Visualisation of Cost Landscapes

Adam Prügel-Bennett
ISIS Research Group
Electronics and Computer Science
University of Southampton

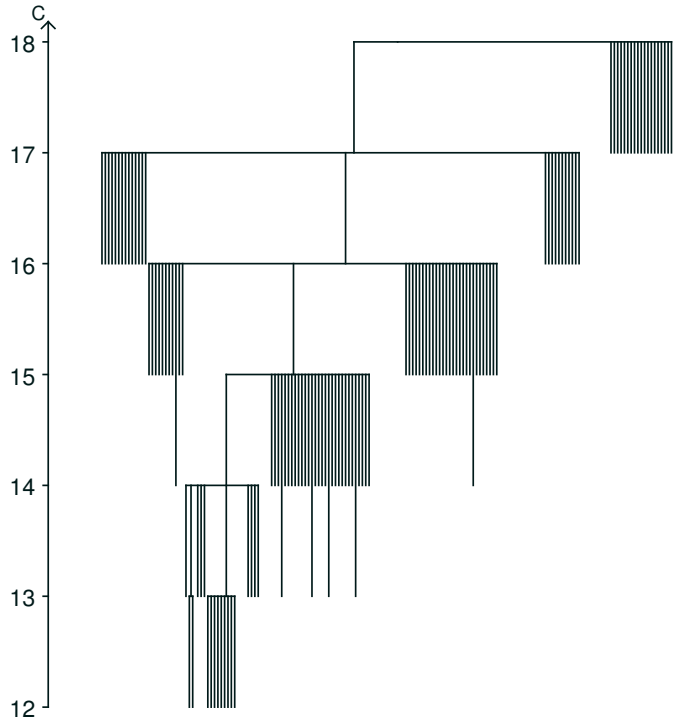
Visualisation of Cost Landscapes

Adam Prügel-Bennett
ISIS Research Group
Electronics and Computer Science
University of Southampton

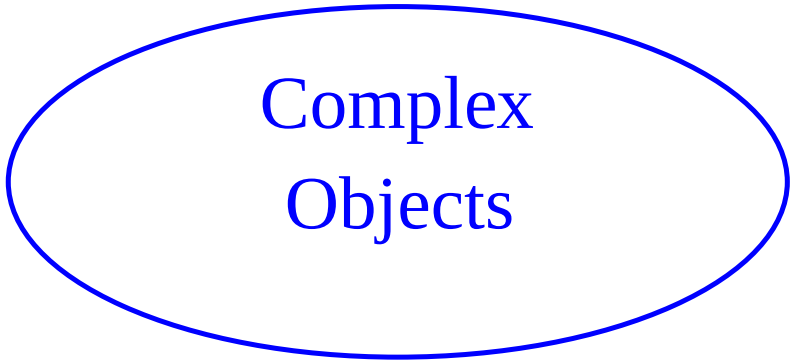
Work in collaboration with Jonathan Hallam and Will Benfold

Outline

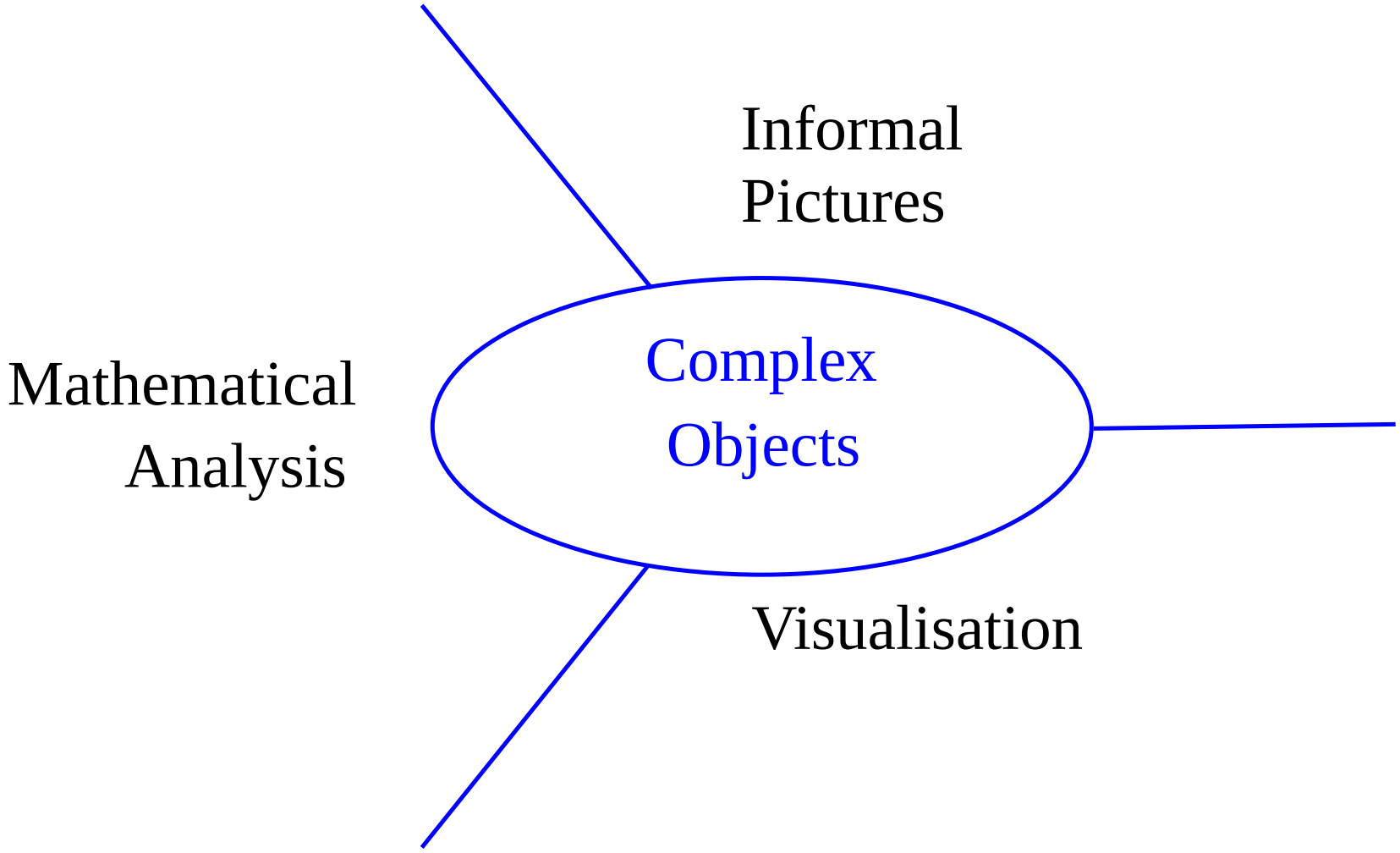
1. Optimisation Problems
2. Cost Landscapes
3. Barrier Trees
4. Example Instances
5. Mapping Configurations
6. Modelling the Problem



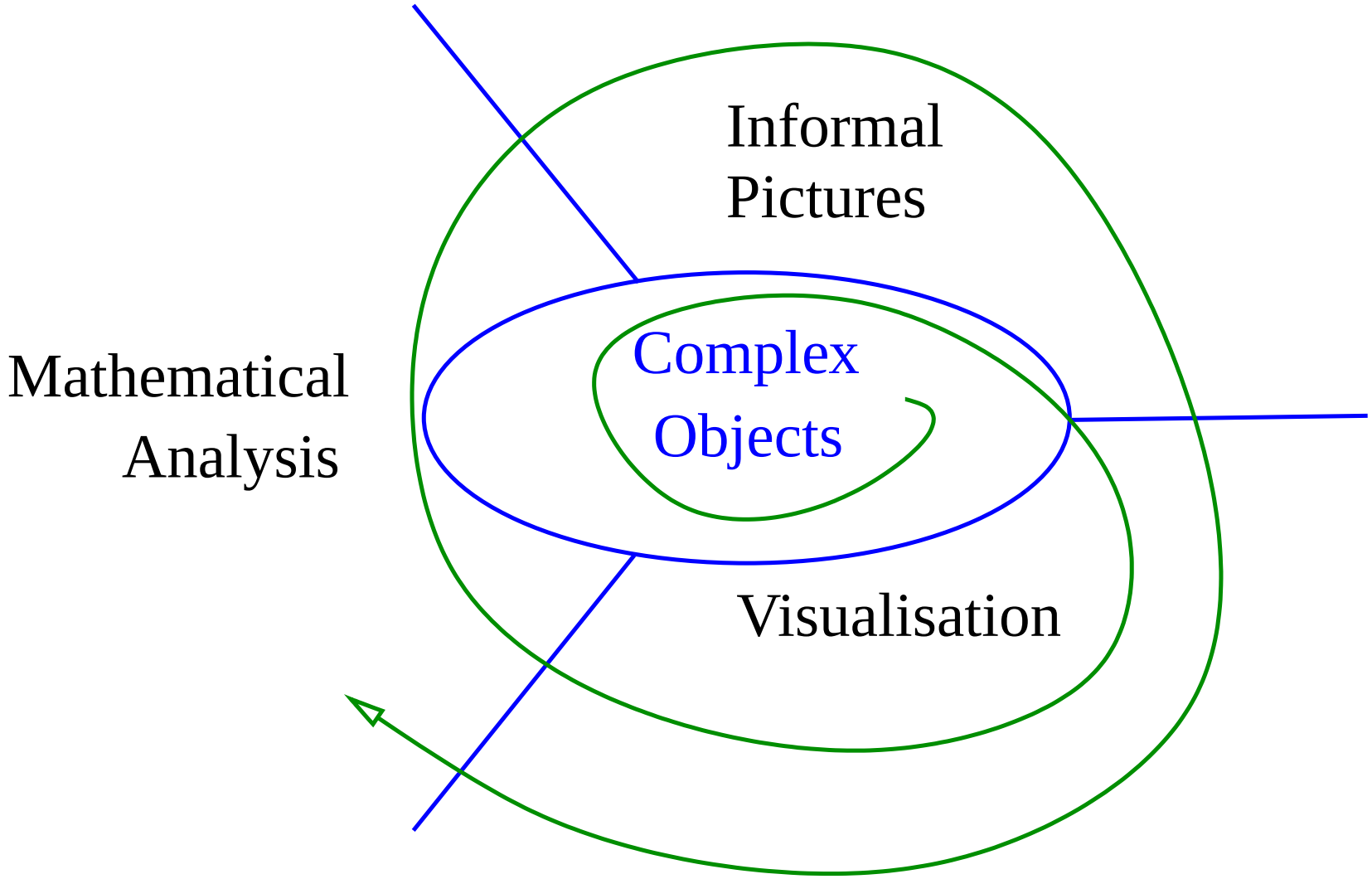
Subtext of Talk



Subtext of Talk



Subtext of Talk



Combinatorial Optimisation Problems

- Set view: $\mathcal{S}$—'the search space'

Combinatorial Optimisation Problems

- Set view: $\mathcal{S}$—'the search space'
- Consider discrete optimisation problems where $\mathcal{S}$ is a finite countable set

Combinatorial Optimisation Problems

- Set view: $\mathcal{S}$—'the search space'
- Consider discrete optimisation problems where $\mathcal{S}$ is a finite countable set
- Cost function: $f : \mathcal{S} \mapsto \mathbb{R}$

Combinatorial Optimisation Problems

- Set view: $\mathcal{S}$—'the search space'
- Consider discrete optimisation problems where $\mathcal{S}$ is a finite countable set
- Cost function: $f : \mathcal{S} \mapsto \mathbb{R}$
- Objective is to find $\boldsymbol{x}^* \in \mathcal{S} : \forall \boldsymbol{y}, f(\boldsymbol{x}^*) \leq f(\boldsymbol{y})$

Combinatorial Optimisation Problems

- Set view: $\mathcal{S}$—'the search space'
- Consider discrete optimisation problems where $\mathcal{S}$ is a finite countable set
- Cost function: $f : \mathcal{S} \mapsto \mathbb{R}$
- Objective is to find $\boldsymbol{x}^* \in \mathcal{S} : \forall \boldsymbol{y}, f(\boldsymbol{x}^*) \leq f(\boldsymbol{y})$
- Usually satisfied with finding $\boldsymbol{x}^* \in \mathcal{S}$: for most $\boldsymbol{y}, f(\boldsymbol{x}^*) \leq f(\boldsymbol{y})$

Combinatorial Optimisation Problems

- Set view: $\mathcal{S}$—'the search space'
- Consider discrete optimisation problems where $\mathcal{S}$ is a finite countable set
- Cost function: $f : \mathcal{S} \mapsto \mathbb{R}$
- Objective is to find $\boldsymbol{x}^* \in \mathcal{S} : \forall \boldsymbol{y}, f(\boldsymbol{x}^*) \leq f(\boldsymbol{y})$
- Usually satisfied with finding $\boldsymbol{x}^* \in \mathcal{S}$: for most $\boldsymbol{y}, f(\boldsymbol{x}^*) \leq f(\boldsymbol{y})$
- Consider mainly degenerate problems $|f(\mathcal{S})| \ll |\mathcal{S}|$

Example: Max-Sat

- Given n Boolean variables $X_i \in \{T, F\}$
- M disjunctive clauses, e.g.

$$c_1 = x_1 \vee \neg x_2 \vee x_3$$
$$c_2 = \neg x_2 \vee x_3 \vee x_5$$
$$\vdots \qquad \vdots$$
$$c_M = x_2 \vee \neg x_4 \vee \neg x_5$$

- Find an assignment, $\boldsymbol{X} \in \{T, F\}^n$ which satisfies the most clauses

Example: Max-Sat

- Given n Boolean variables $X_i \in \{T, F\}$
- M disjunctive clauses, e.g.

$$c_1 = x_1 \vee \neg x_2 \vee x_3$$
$$c_2 = \neg x_2 \vee x_3 \vee x_5$$
$$\vdots \qquad \vdots$$
$$c_M = x_2 \vee \neg x_4 \vee \neg x_5$$

- Find an assignment, $\boldsymbol{X} \in \{T, F\}^n$ which satisfies the most clauses

Example: Max-Sat

- Given n Boolean variables $X_i \in \{T, F\}$
- M disjunctive clauses, e.g.

$$c_1 = x_1 \vee \neg x_2 \vee x_3$$
$$c_2 = \neg x_2 \vee x_3 \vee x_5$$
$$\vdots \qquad \vdots$$
$$c_M = x_2 \vee \neg x_4 \vee \neg x_5$$

- Find an assignment, $\boldsymbol{X} \in \{T, F\}^n$ which satisfies the most clauses

Max-Sat Problems

- Arises naturally in digital circuits
- Many planning problems reduce to Max-Sat
- Any non-deterministic Turing machine can be reduced to a SAT problem—Cook's theorem
- Factorisation can be solved using a SAT solver

Max-Sat Problems

- Arises naturally in digital circuits
- Many planning problems reduce to Max-Sat
- Any non-deterministic Turing machine can be reduced to a SAT problem—Cook's theorem
- Factorisation can be solved using a SAT solver

Max-Sat Problems

- Arises naturally in digital circuits
- Many planning problems reduce to Max-Sat
- Any non-deterministic Turing machine can be reduced to a SAT problem—Cook's theorem
- Factorisation can be solved using a SAT solver

Max-Sat Problems

- Arises naturally in digital circuits
- Many planning problems reduce to Max-Sat
- Any non-deterministic Turing machine can be reduced to a SAT problem—Cook's theorem
- Factorisation can be solved using a SAT solver

Max-Sat Problems

- Arises naturally in digital circuits
- Many planning problems reduce to Max-Sat
- Any non-deterministic Turing machine can be reduced to a SAT problem—Cook's theorem
- Factorisation can be solved using a SAT solver

Machine Learning Example

- Perceptron is a classic learning machine for performing binary classification

- Given a data set $\mathcal{D} = \{\boldsymbol{x}_i, c_i\}_{i=1}^{P}$

  - $\boldsymbol{x}_i$—n-dimensional input pattern
  - $c_i \in \{-1, 1\}$—class label

- Find a weight vector $\boldsymbol{w}$ such that

$$c_i \, \boldsymbol{w}^T \boldsymbol{x}_i > 0$$

  for as many patterns as possible

Machine Learning Example

- Perceptron is a classic learning machine for performing binary classification

- Given a data set $\mathcal{D} = \{\boldsymbol{x}_i, c_i\}_{i=1}^{P}$

  - $\boldsymbol{x}_i$—n-dimensional input pattern
  - $c_i \in \{-1, 1\}$—class label

- Find a weight vector $\boldsymbol{w}$ such that

$$c_i \, \boldsymbol{w}^T \boldsymbol{x}_i > 0$$

  for as many patterns as possible

Machine Learning Example

- Perceptron is a classic learning machine for performing binary classification

- Given a data set $\mathcal{D} = \{\boldsymbol{x}_i, c_i\}_{i=1}^{P}$

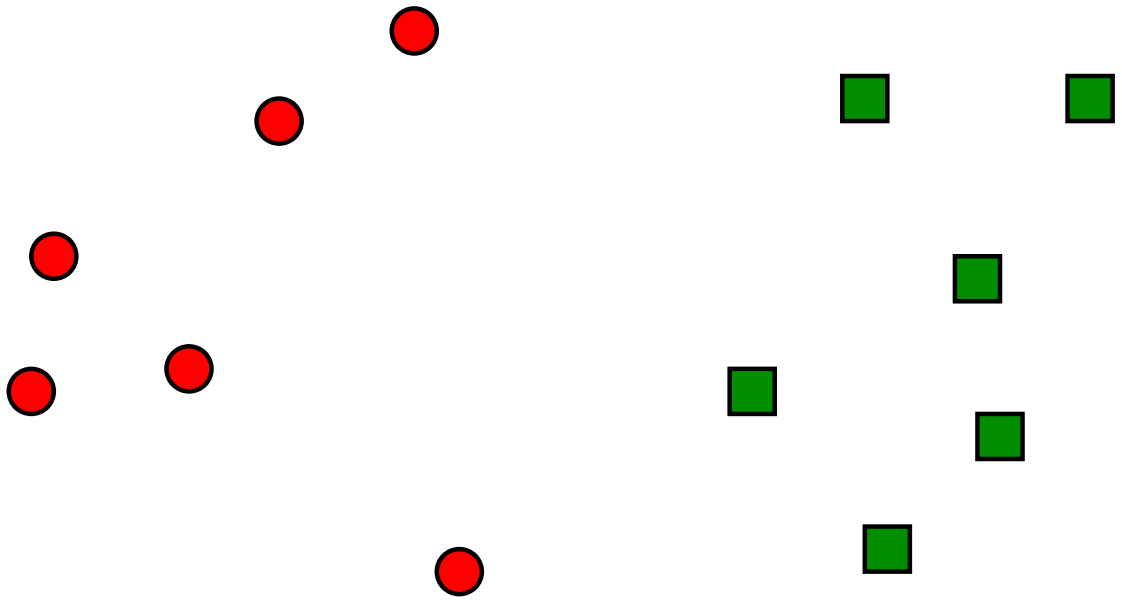
  - $\boldsymbol{x}_i$—n-dimensional input pattern
  - $c_i \in \{-1, 1\}$—class label

- Find a weight vector $\boldsymbol{w}$ such that

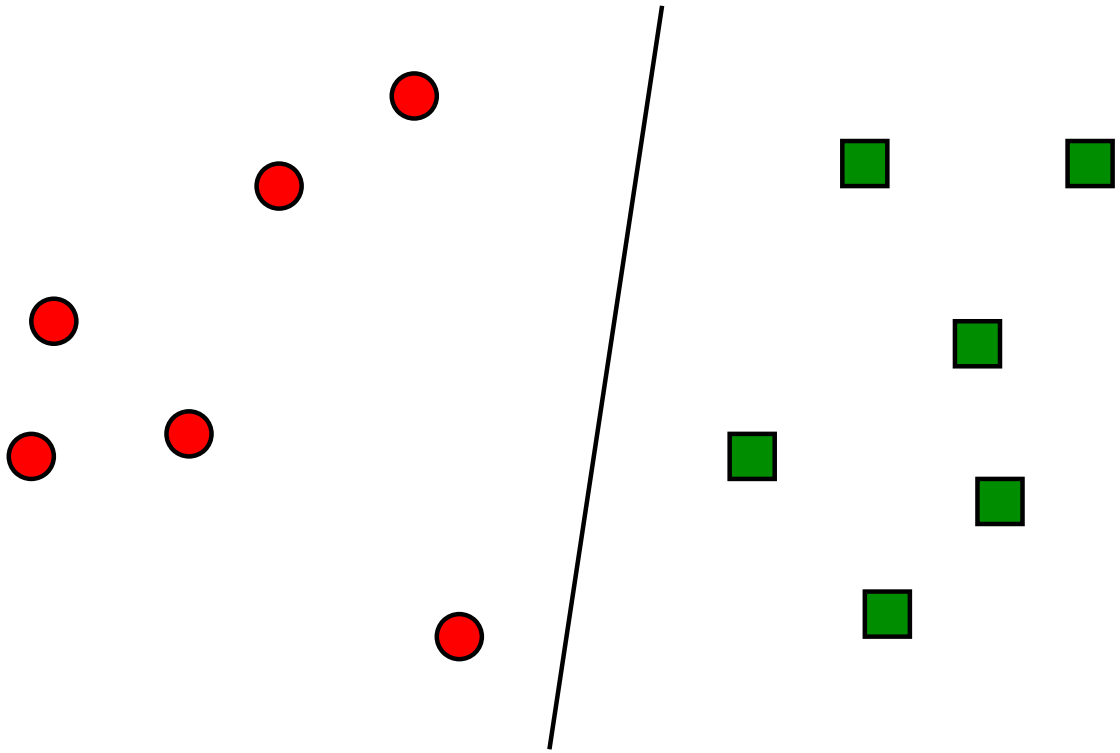
$$c_i \, \boldsymbol{w}^T \boldsymbol{x}_i > 0$$

  for as many patterns as possible

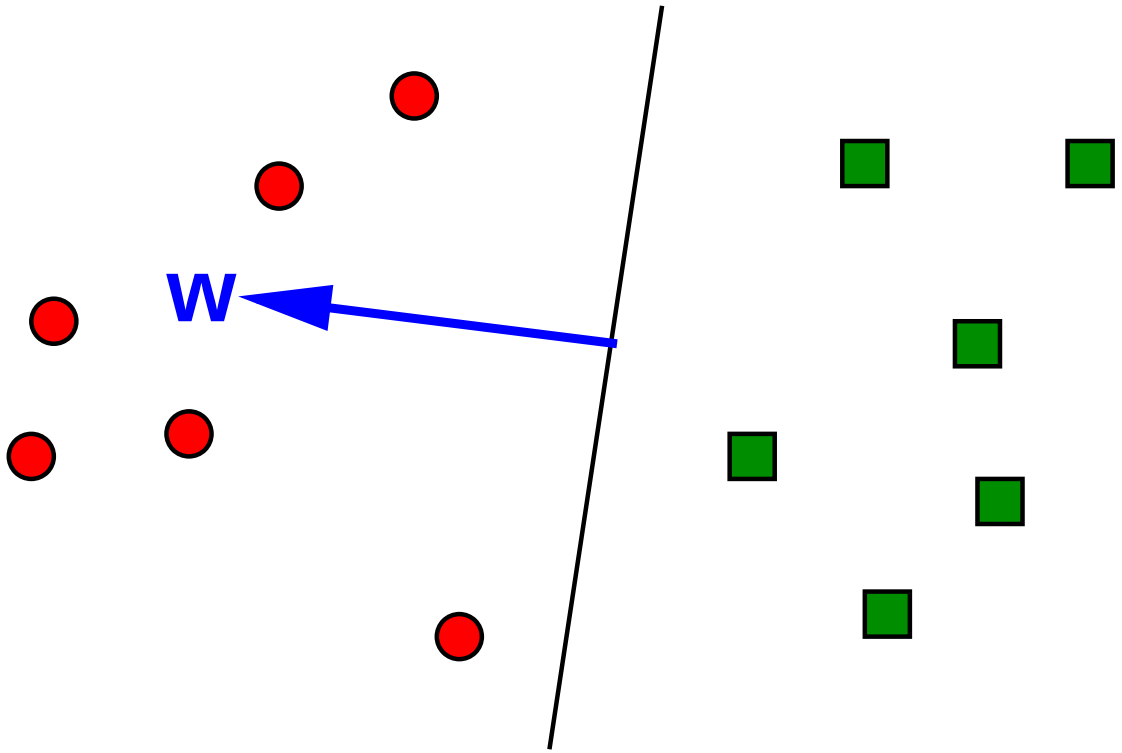
Separating Plane



Separating Plane



Separating Plane



Ising Perceptron

- Weight vector is strongly constrained $\boldsymbol{w} = \{-1, 1\}^n$
- Problem becomes NP-Hard (integer programming problem)
- Both Max-Sat and the Ising perceptron have a lot of structure not captured by the set view of optimisation problems

Ising Perceptron

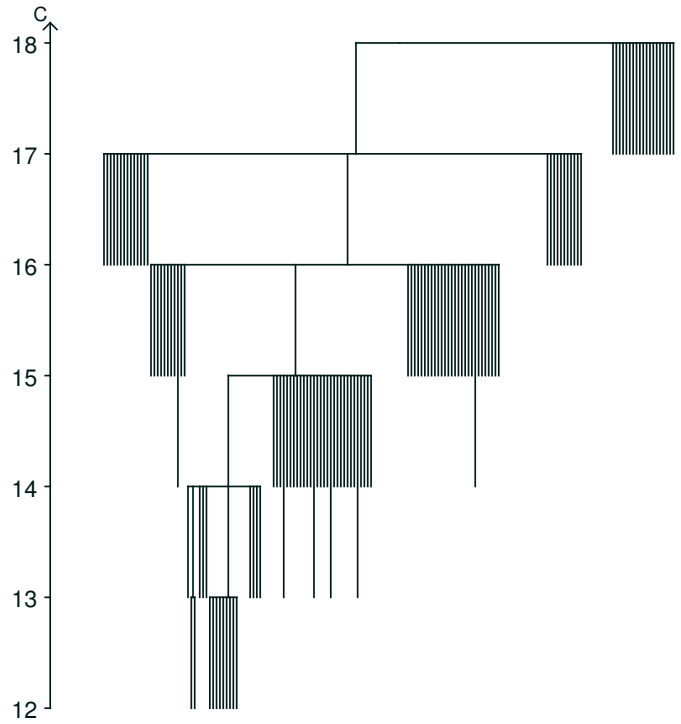
- Weight vector is strongly constrained $\boldsymbol{w} = \{-1, 1\}^n$
- Problem becomes NP-Hard (integer programming problem)
- Both Max-Sat and the Ising perceptron have a lot of structure not captured by the set view of optimisation problems

Ising Perceptron

- Weight vector is strongly constrained $\boldsymbol{w} = \{-1, 1\}^n$
- Problem becomes NP-Hard (integer programming problem)
- Both Max-Sat and the Ising perceptron have a lot of structure not captured by the set view of optimisation problems

Outline

1. Optimisation Problems
2. Cost Landscapes
3. Barrier Trees
4. Example Instances
5. Mapping Configurations
6. Modelling the Problem



Neighbourhood

- We can introduce a topology to a search space by defining a neighbourhood function $\mathcal{N} : \mathcal{S} \mapsto 2^{\mathcal{S}}$

- Neighbourhoods are often connected with local search operators

  $\mathcal{N}(\boldsymbol{x})$ is the set of configurations we can move to from $\boldsymbol{x}$

- Key Heuristic: Neighbouring configurations are more likely to have a similar cost than random configurations

Neighbourhood

- We can introduce a topology to a search space by defining a neighbourhood function $\mathcal{N} : \mathcal{S} \mapsto 2^{\mathcal{S}}$

- Neighbourhoods are often connected with local search operators

  $\mathcal{N}(\boldsymbol{x})$ is the set of configurations we can move to from $\boldsymbol{x}$

- Key Heuristic: Neighbouring configurations are more likely to have a similar cost than random configurations

Neighbourhood

- We can introduce a topology to a search space by defining a neighbourhood function $\mathcal{N} : \mathcal{S} \mapsto 2^{\mathcal{S}}$

- Neighbourhoods are often connected with local search operators

  $\mathcal{N}(\boldsymbol{x})$ is the set of configurations we can move to from $\boldsymbol{x}$

- Key Heuristic: Neighbouring configurations are more likely to have a similar cost than random configurations

Neighbourhood

- We can introduce a topology to a search space by defining a neighbourhood function $\mathcal{N} : \mathcal{S} \mapsto 2^{\mathcal{S}}$

- Neighbourhoods are often connected with local search operators

  $\mathcal{N}(\boldsymbol{x})$ is the set of configurations we can move to from $\boldsymbol{x}$

- Key Heuristic: Neighbouring configurations are more likely to have a similar cost than random configurations

Hamming Neighbourhood

- A natural neighbourhood for binary strings is the Hamming neighbourhood
- Configurations differing at a single site are neighbours

$$\boldsymbol{X} = (T, T, F, T, F, T, T) \qquad \boldsymbol{w} = (+1, -1, -1, +1, +1, +1)$$
$$\boldsymbol{X} = (T, F, F, T, F, T, T) \qquad \boldsymbol{w} = (+1, -1, -1, +1, -1, +1)$$

- Changing one variable in Max-Sat shouldn't affect too many clauses
- Changing a weight in the Ising perceptron causes a minimum shift in the separating plane

Hamming Neighbourhood

- A natural neighbourhood for binary strings is the Hamming neighbourhood
- Configurations differing at a single site are neighbours

$$\boldsymbol{X} = (T, T, F, T, F, T, T) \qquad \boldsymbol{w} = (+1, -1, -1, +1, +1, +1)$$
$$\boldsymbol{X} = (T, F, F, T, F, T, T) \qquad \boldsymbol{w} = (+1, -1, -1, +1, -1, +1)$$

- Changing one variable in Max-Sat shouldn't affect too many clauses
- Changing a weight in the Ising perceptron causes a minimum shift in the separating plane

Hamming Neighbourhood

- A natural neighbourhood for binary strings is the Hamming neighbourhood
- Configurations differing at a single site are neighbours

$$\boldsymbol{X} = (T, T, F, T, F, T, T) \qquad \boldsymbol{w} = (+1, -1, -1, +1, +1, +1)$$
$$\boldsymbol{X} = (T, F, F, T, F, T, T) \qquad \boldsymbol{w} = (+1, -1, -1, +1, -1, +1)$$

- Changing one variable in Max-Sat shouldn't affect too many clauses
- Changing a weight in the Ising perceptron causes a minimum shift in the separating plane

Hamming Neighbourhood

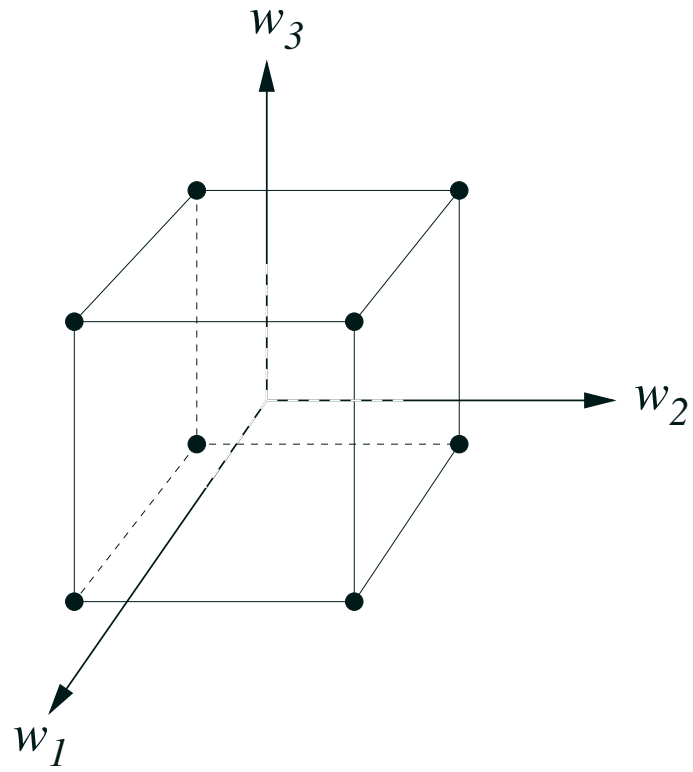
- A natural neighbourhood for binary strings is the Hamming neighbourhood
- Configurations differing at a single site are neighbours

$$\boldsymbol{X} = (T, T, F, T, F, T, T) \qquad \boldsymbol{w} = (+1, -1, -1, +1, +1, +1)$$
$$\boldsymbol{X} = (T, F, F, T, F, T, T) \qquad \boldsymbol{w} = (+1, -1, -1, +1, -1, +1)$$

- Changing one variable in Max-Sat shouldn't affect too many clauses
- Changing a weight in the Ising perceptron causes a minimum shift in the separating plane

Hypercube Topology

- The Hamming neighbourhood induces a hypercube topology on the search space of binary strings



3D Binary String Search Space

Other Neighbourhoods

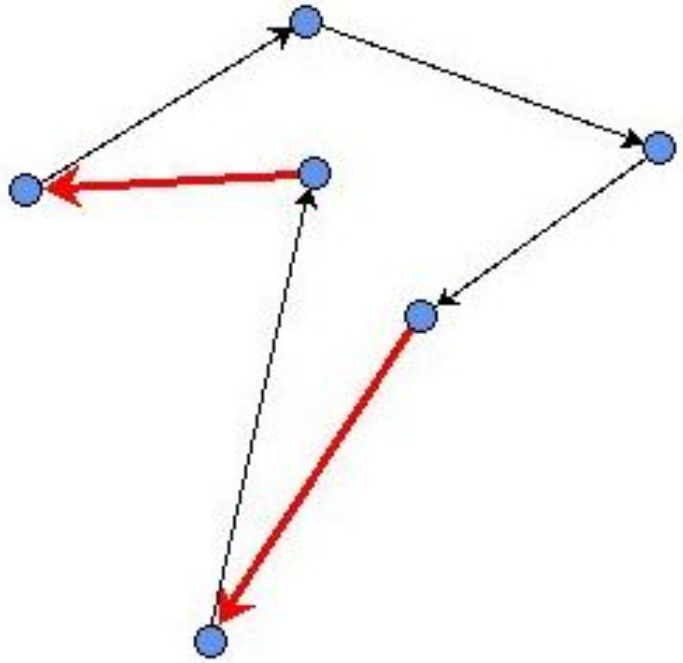
- For binary string problems the Hamming neighbourhood is so natural it is difficult to image anything else

Other Neighbourhoods

- For binary string problems the Hamming neighbourhood is so natural it is difficult to image anything else
- But what is the neighbourhood for TSP?

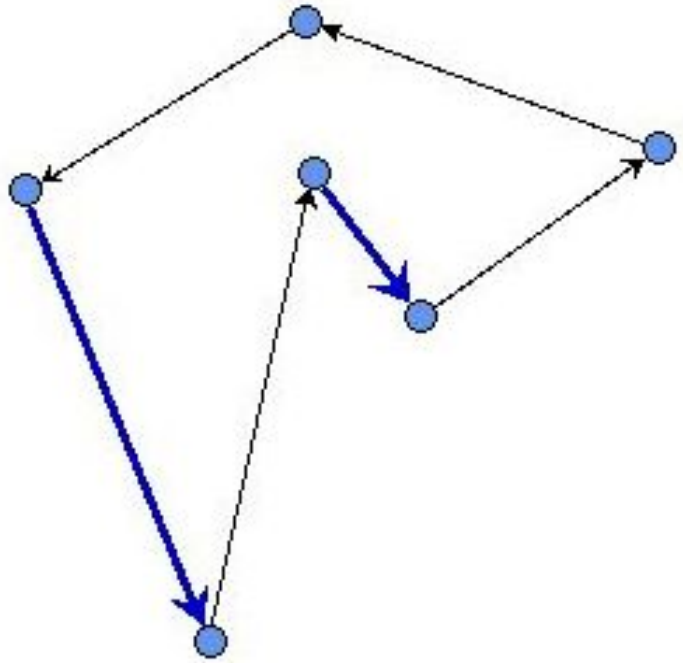
Other Neighbourhoods

- For binary string problems the Hamming neighbourhood is so natural it is difficult to image anything else
- But what is the neighbourhood for TSP?



Other Neighbourhoods

- For binary string problems the Hamming neighbourhood is so natural it is difficult to image anything else
- But what is the neighbourhood for TSP?



2-Opt neighbours

Cost Landscape

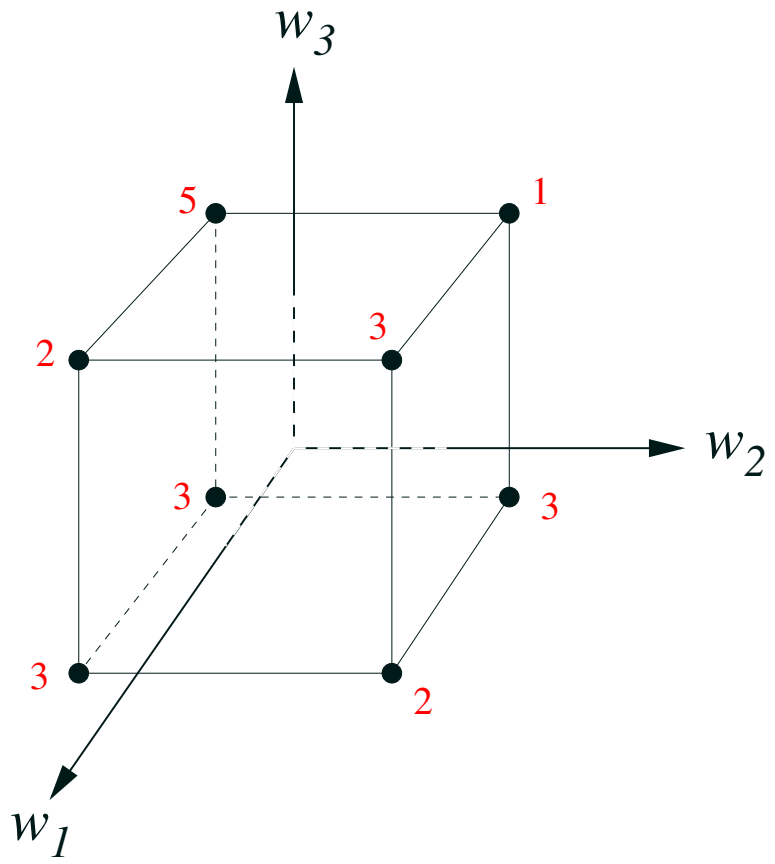
- Cost function + Neighbourhood = Landscape

Cost Landscape

- Cost function + Neighbourhood = Landscape

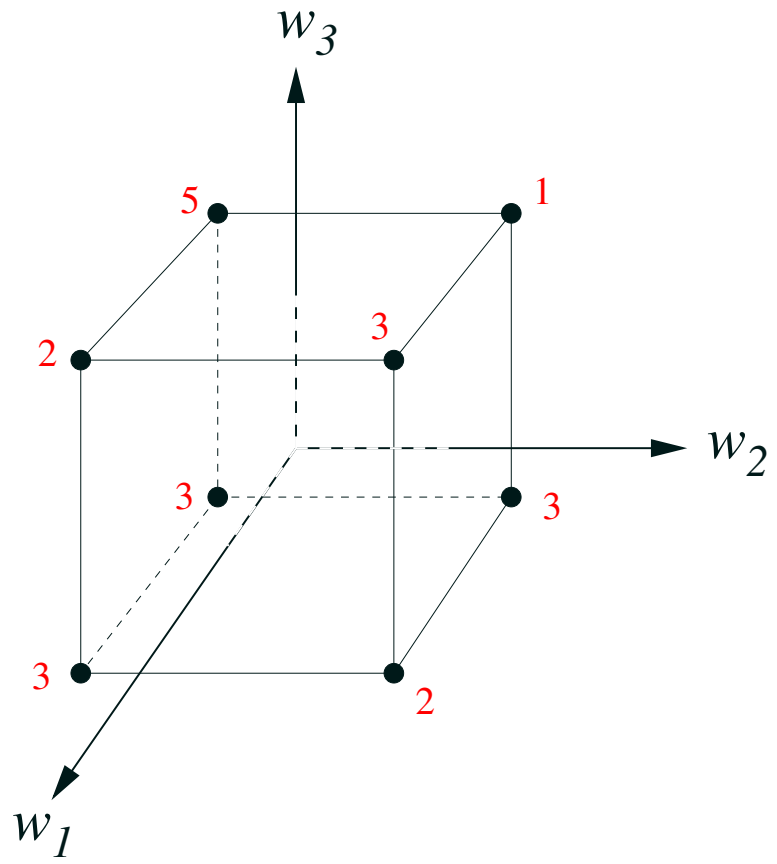
Cost Landscape

- Cost function + Neighbourhood = Landscape



Cost Landscape

- Cost function + Neighbourhood = Landscape



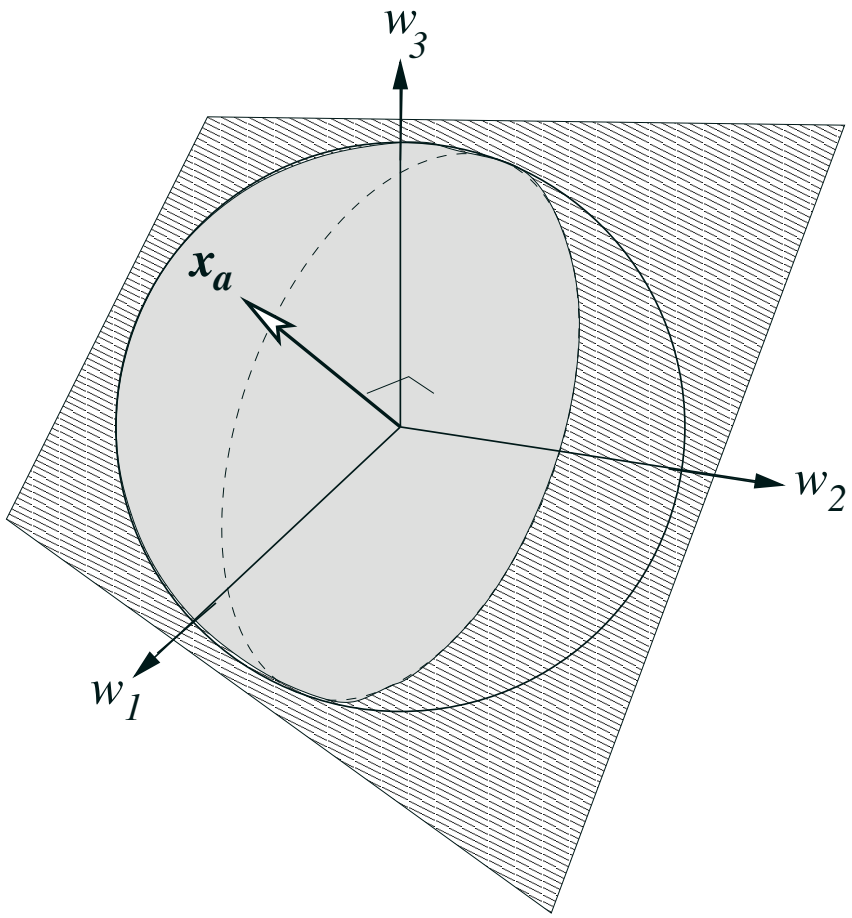
- Unfortunately difficult to visualise in high dimensions

Costs in the Ising Perceptron

Each input divides the search space in two

Costs in the Ising Perceptron

Each input divides the search space in two

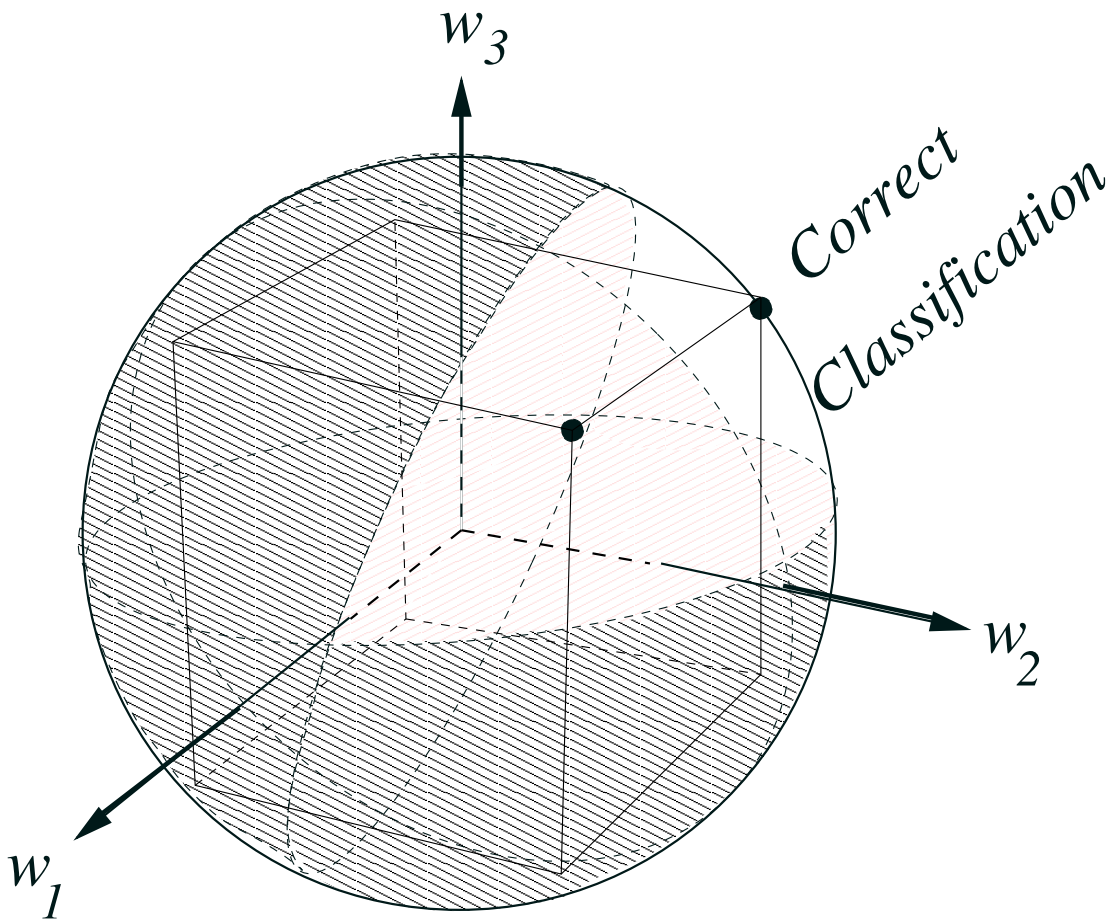


Correctly Classified Region

The set of inputs defines a cone on the hypersphere which correctly classifies all inputs

Correctly Classified Region

The set of inputs defines a cone on the hypersphere which correctly classifies all inputs

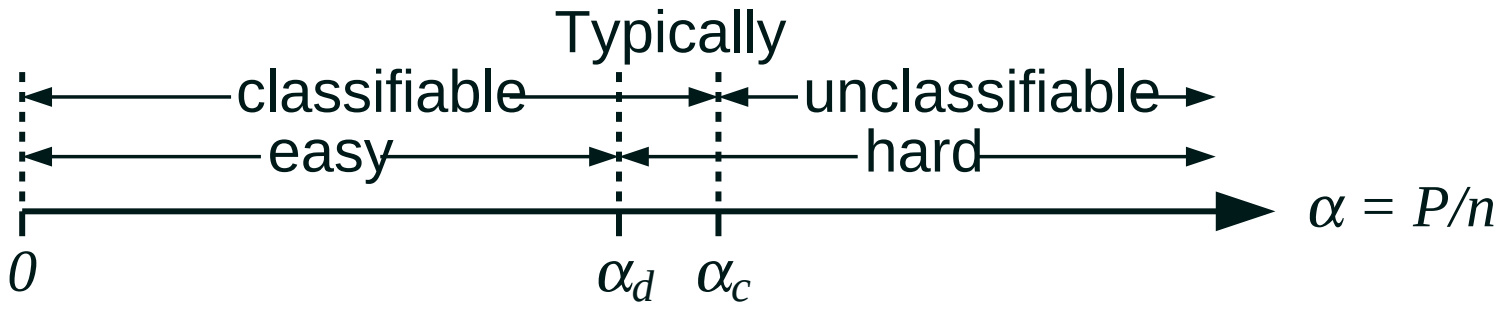


Increasing the Problem Difficulty

- The problem difficulty is increased as we increase $\alpha = P/n$

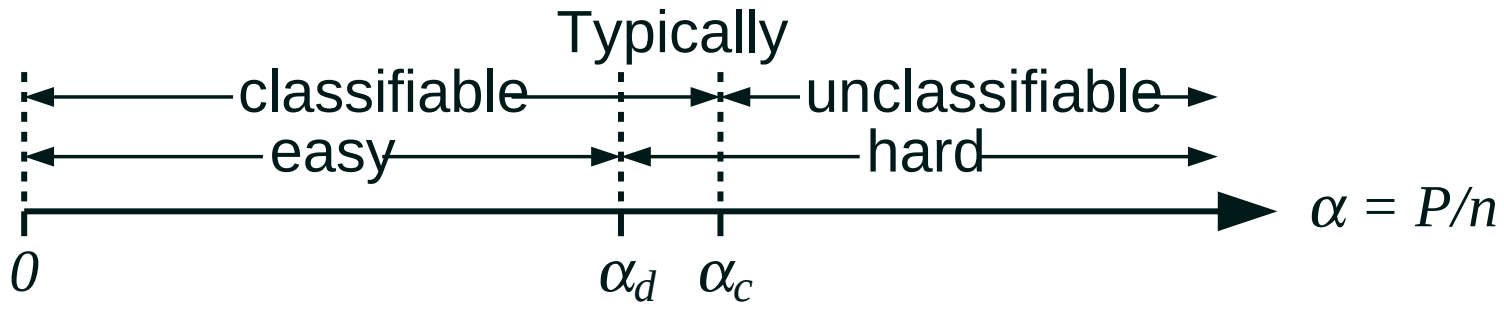
Increasing the Problem Difficulty

- The problem difficulty is increased as we increase $\alpha = P/n$



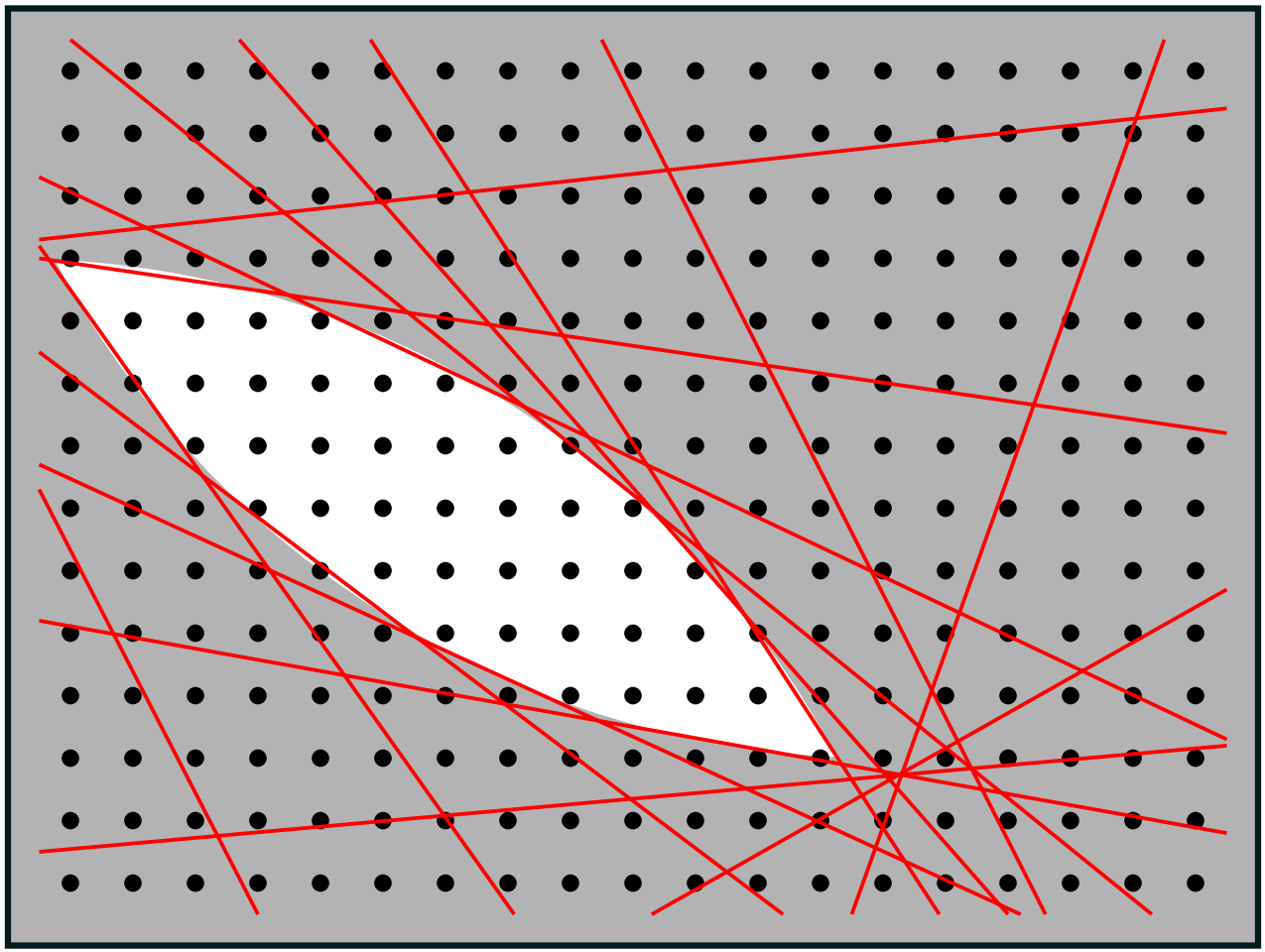
Increasing the Problem Difficulty

- The problem difficulty is increased as we increase $\alpha = P/n$



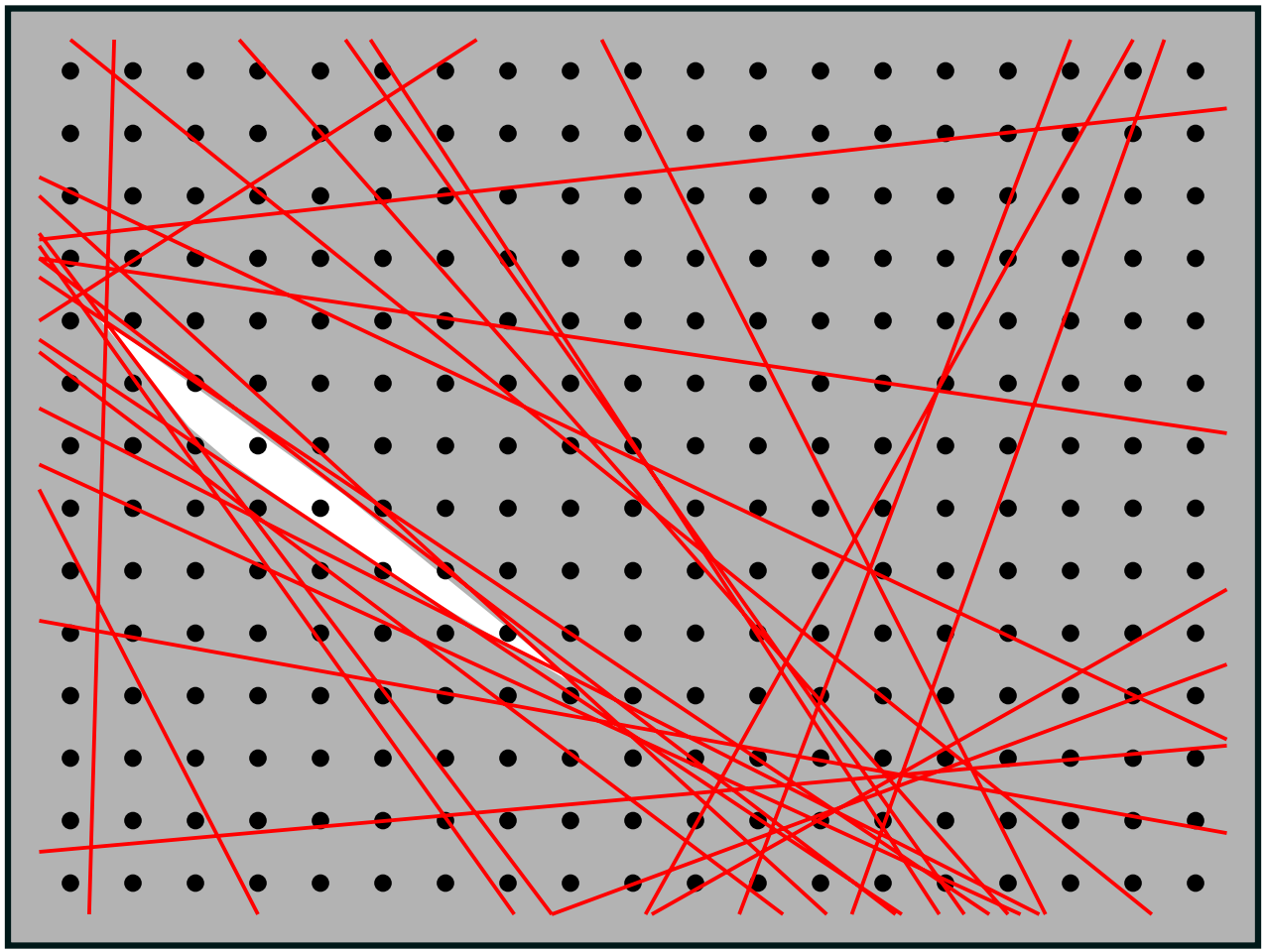
- Typically means that the exceptions become exponentially rare with n

Schematic representation of low α

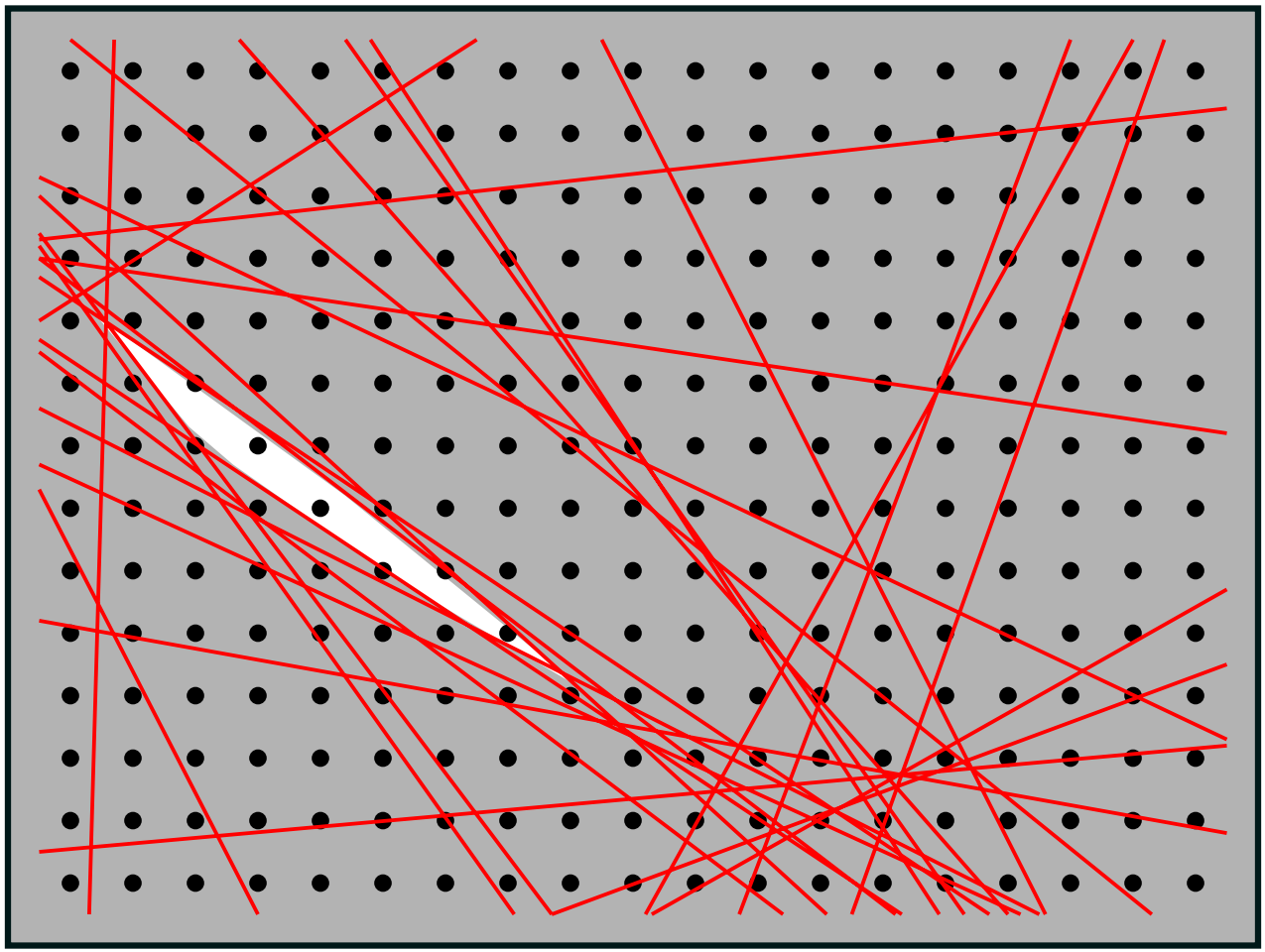


Note that the real search space is high dimensional

Schematic representation at higher α



Schematic representation at higher α



Solutions are typically disconnected

Replica Symmetry Breaking

- The problem becomes hard just as the solutions become disconnected
- The solutions can be a long way apart
- The solution space "shatters" into many local optima rather like a piece of glass
  - ⋆ shattering is random
  - ⋆ there are (exponentially) many local optima
  - ⋆ the sizes of the local optima differ dramatically
- In calculations this shattering causes "replica-symmetry breaking"

Replica Symmetry Breaking

- The problem becomes hard just as the solutions become disconnected
- The solutions can be a long way apart
- The solution space "shatters" into many local optima rather like a piece of glass
  - ⋆ shattering is random
  - ⋆ there are (exponentially) many local optima
  - ⋆ the sizes of the local optima differ dramatically
- In calculations this shattering causes "replica-symmetry breaking"

Replica Symmetry Breaking

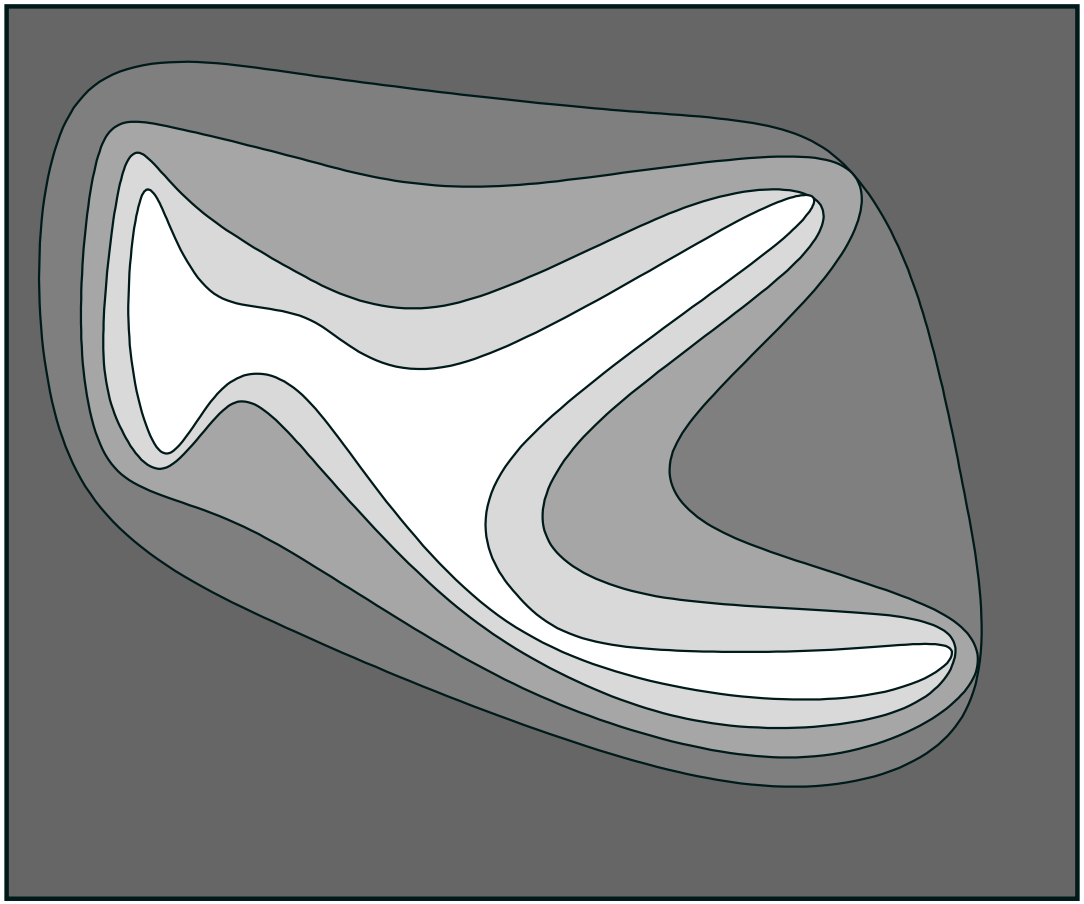
- The problem becomes hard just as the solutions become disconnected
- The solutions can be a long way apart
- The solution space "shatters" into many local optima rather like a piece of glass
  - ⋆ shattering is random
  - ⋆ there are (exponentially) many local optima
  - ⋆ the sizes of the local optima differ dramatically
- In calculations this shattering causes "replica-symmetry breaking"

Replica Symmetry Breaking

- The problem becomes hard just as the solutions become disconnected
- The solutions can be a long way apart
- The solution space "shatters" into many local optima rather like a piece of glass
  - ⋆ shattering is random
  - ⋆ there are (exponentially) many local optima
  - ⋆ the sizes of the local optima differ dramatically
- In calculations this shattering causes "replica-symmetry breaking"

Easy Phase

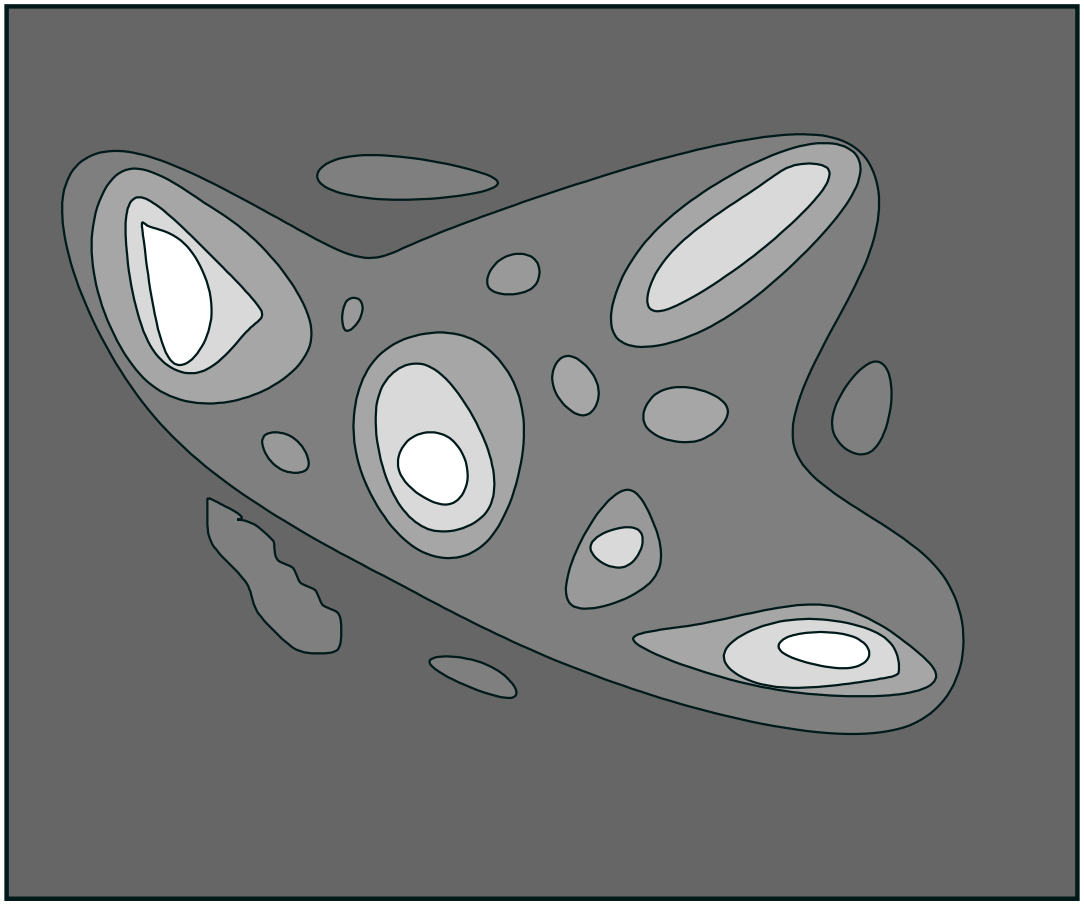
$\alpha < \alpha_d$



'Gradient' leads to global optima

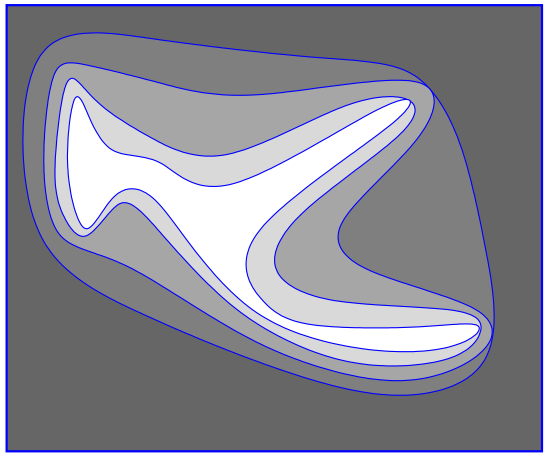
Hard Phase

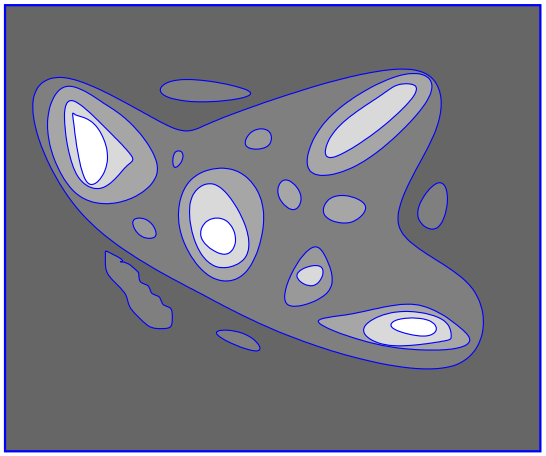
$\alpha > \alpha_d$



Ambiguous 'gradients'

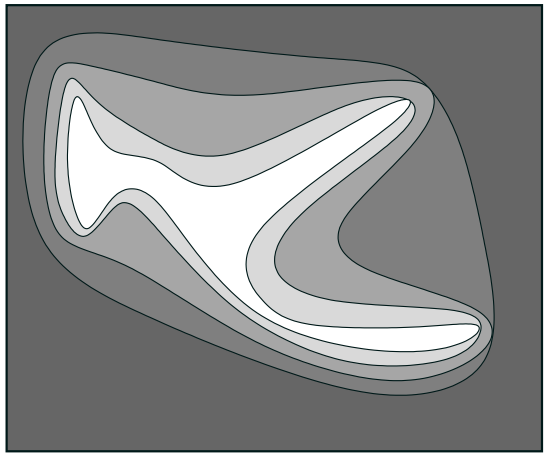
Is this an Accurate Picture?

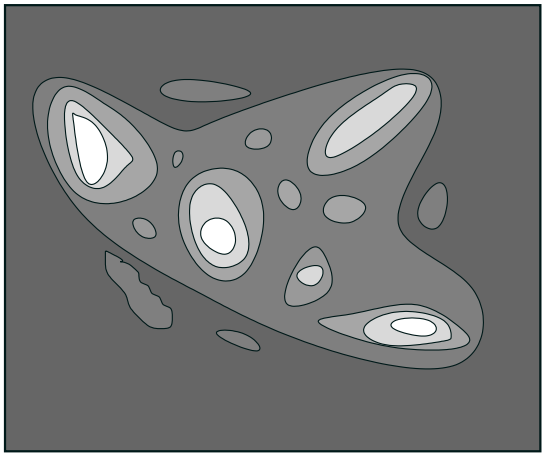




- Interested in high dimensional discrete problems
- Usually there are exponential number of local optima
- Phase transition can be more complicated (e.g. number partitioning problem)

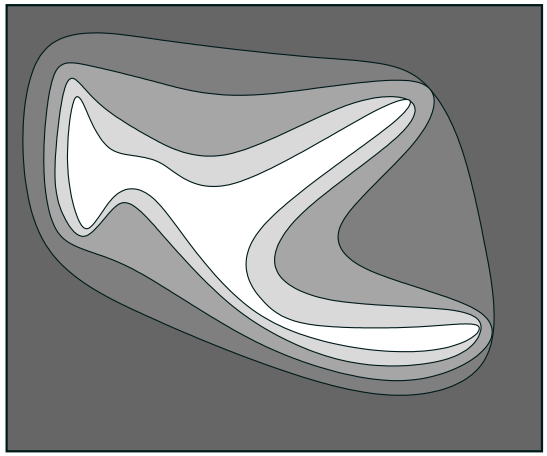
Is this an Accurate Picture?

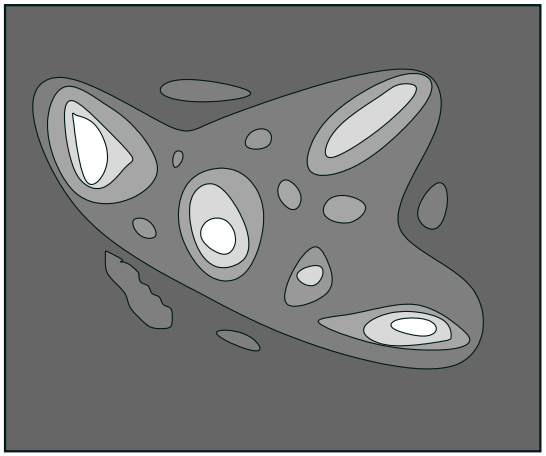




- Interested in high dimensional discrete problems
- Usually there are exponential number of local optima
- Phase transition can be more complicated (e.g. number partitioning problem)

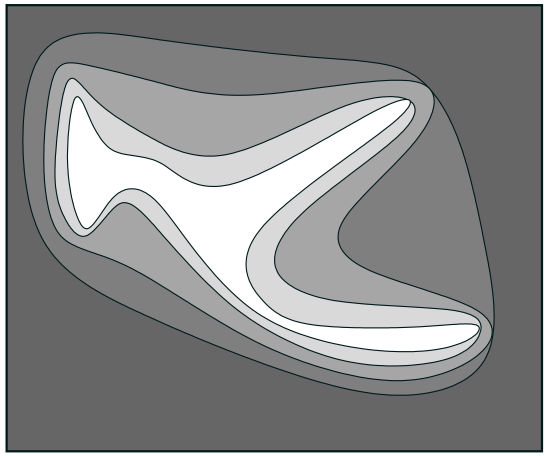
Is this an Accurate Picture?

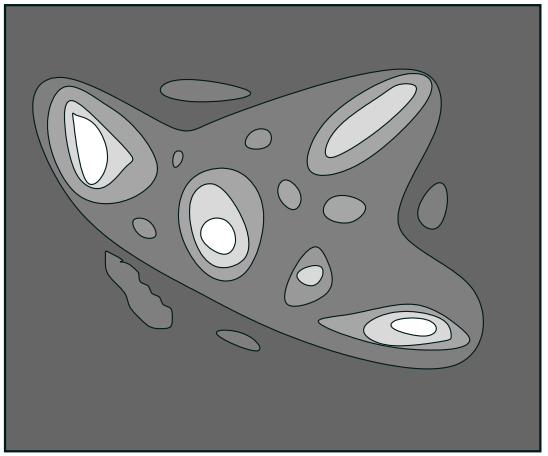




- Interested in high dimensional discrete problems
- Usually there are exponential number of local optima
- Phase transition can be more complicated (e.g. number partitioning problem)

Is this an Accurate Picture?

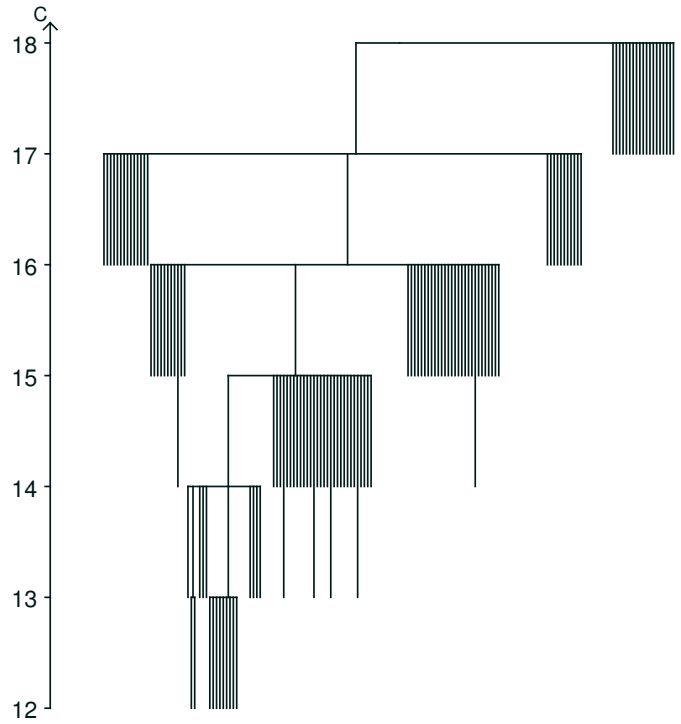




- Interested in high dimensional discrete problems
- Usually there are exponential number of local optima
- Phase transition can be more complicated (e.g. number partitioning problem)

Outline

1. Optimisation Problems
2. Cost Landscapes
3. Barrier Trees
4. Example Instances
5. Mapping Configurations
6. Modelling the Problem



Barrier Trees

- To help visualise the search space we can build a 'Barrier Tree'
- Leaves of the tree are local minima
- Merging vertices correspond to lowest cost saddle-points
- Height of vertices indicate fitness

Barrier Trees

- To help visualise the search space we can build a 'Barrier Tree'
- Leaves of the tree are local minima
- Merging vertices correspond to lowest cost saddle-points
- Height of vertices indicate fitness

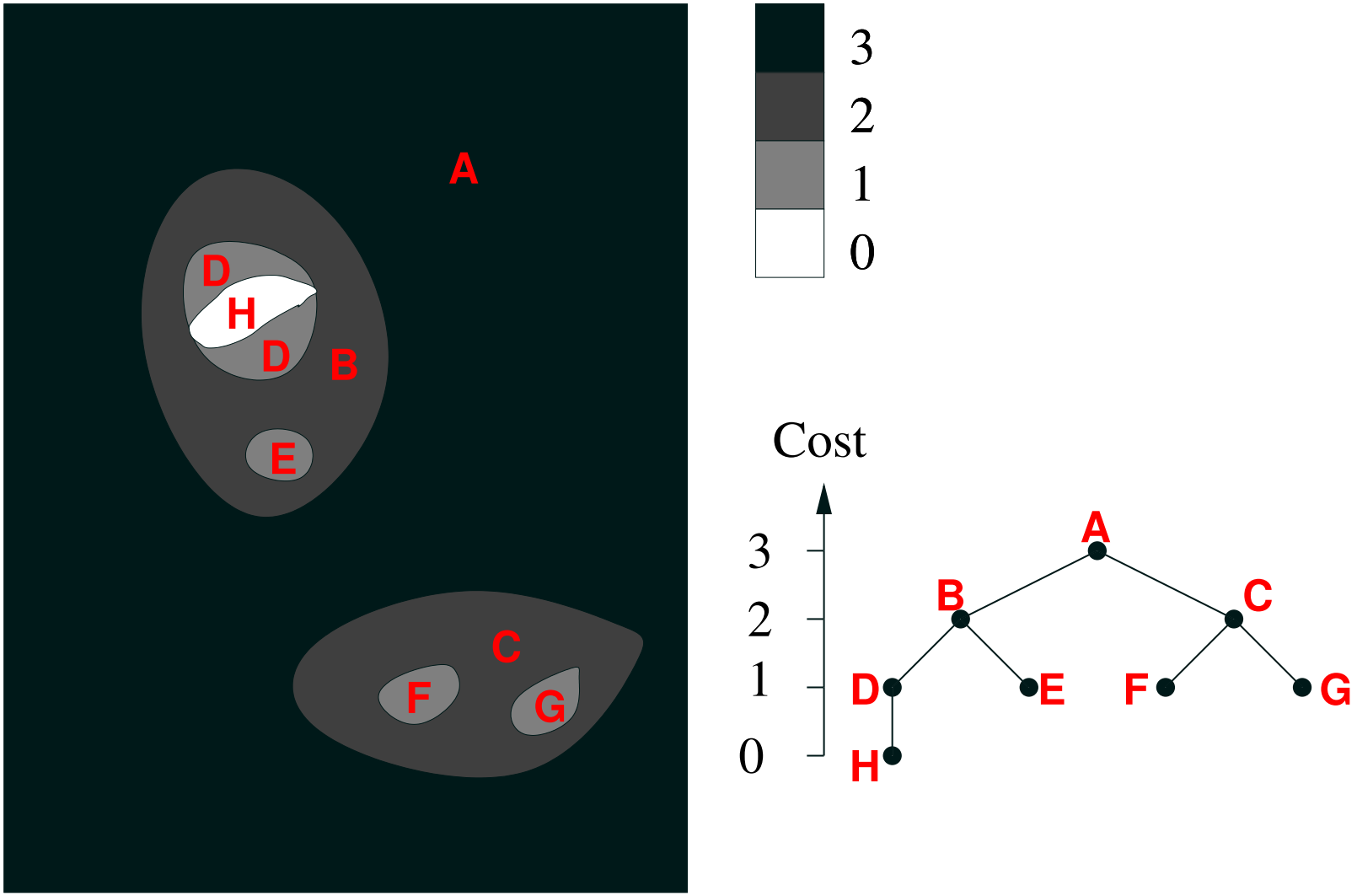
Barrier Trees

- To help visualise the search space we can build a 'Barrier Tree'
- Leaves of the tree are local minima
- Merging vertices correspond to lowest cost saddle-points
- Height of vertices indicate fitness

Barrier Trees

- To help visualise the search space we can build a 'Barrier Tree'
- Leaves of the tree are local minima
- Merging vertices correspond to lowest cost saddle-points
- Height of vertices indicate fitness

Example



Discrete Search Spaces

- To formalise these notion we define a path
  - ⋆ $\pi(\boldsymbol{x}, \boldsymbol{y}) = \left(\boldsymbol{x_i} \middle| \boldsymbol{x}_1 = \boldsymbol{x} \wedge \boldsymbol{x}_n = \boldsymbol{y} \wedge \boldsymbol{x}_{i+1} \in \mathcal{N}(\boldsymbol{x}_i)\right)$
  - ⋆ A sequence of Neighbouring configurations from $\boldsymbol{x}$ to $\boldsymbol{y}$
- Assume that the Neighbourhood relation is symmetric

Discrete Search Spaces

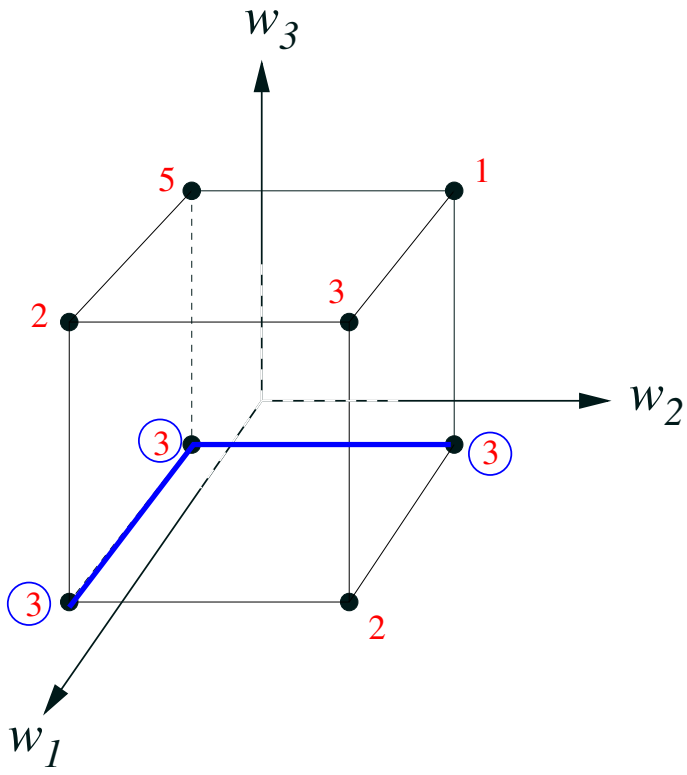
- To formalise these notion we define a path
  - ⋆ $\pi(\boldsymbol{x}, \boldsymbol{y}) = \left(\boldsymbol{x_i} \middle| \boldsymbol{x}_1 = \boldsymbol{x} \wedge \boldsymbol{x}_n = \boldsymbol{y} \wedge \boldsymbol{x}_{i+1} \in \mathcal{N}(\boldsymbol{x}_i)\right)$
  - ⋆ A sequence of Neighbouring configurations from $\boldsymbol{x}$ to $\boldsymbol{y}$

- Assume that the Neighbourhood relation is symmetric

Level Connected Sets

- We define the level connectedness by the equivalence relation

$$\mathcal{LC} = \{(\boldsymbol{x}, \boldsymbol{y}) | \exists \pi(\boldsymbol{x}, \boldsymbol{y}) \wedge \forall \boldsymbol{x}_i \in \pi(\boldsymbol{x}, \boldsymbol{y}), f(\boldsymbol{x}_i) = f(\boldsymbol{x})\}$$



Limitation of Level Connectedness

- There are often a very large number of level-connected sets
- Consider the 'ones-max' problem where

$$f(\boldsymbol{x}) = \text{number of ones in } \boldsymbol{x}$$

- Every configuration is a level-connected set with a Hamming neighbourhood

Limitation of Level Connectedness

- There are often a very large number of level-connected sets
- Consider the 'ones-max' problem where

$$f(\boldsymbol{x}) = \text{number of ones in } \boldsymbol{x}$$

- Every configuration is a level-connected set with a Hamming neighbourhood

Limitation of Level Connectedness

- There are often a very large number of level-connected sets
- Consider the 'ones-max' problem where

$$f(\boldsymbol{x}) = \text{number of ones in } \boldsymbol{x}$$

- Every configuration is a level-connected set with a Hamming neighbourhood

Level Accessible

- We define accessibility as the relation

$$\mathcal{A} = \{(\boldsymbol{x}, \boldsymbol{y}) | \exists \pi(\boldsymbol{x}, \boldsymbol{y}) \wedge \forall \boldsymbol{x}_i \in \pi(\boldsymbol{x}, \boldsymbol{y}), f(\boldsymbol{x}_i) \leq f(\boldsymbol{x})\}$$

  i.e. there exists a path with no configuration exceeding the cost of the initial configuration

- Level accessibility is the equivalence relation that

$$\mathcal{LA} = \{(\boldsymbol{x}, \boldsymbol{y}) | (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{A} \wedge f(\boldsymbol{x}) = f(\boldsymbol{y})\}$$

- Level accessibility induces a partitioning of the search space into level-accessible sets $\mathcal{S} = \bigcup_{i=1} \mathcal{V}_i$

- In ones-max there are $N + 1$ level accessible sets

Level Accessible

- We define accessibility as the relation

$$\mathcal{A} = \{(\boldsymbol{x}, \boldsymbol{y}) | \exists \pi(\boldsymbol{x}, \boldsymbol{y}) \wedge \forall \boldsymbol{x}_i \in \pi(\boldsymbol{x}, \boldsymbol{y}), f(\boldsymbol{x}_i) \leq f(\boldsymbol{x})\}$$

  i.e. there exists a path with no configuration exceeding the cost of the initial configuration

- Level accessibility is the equivalence relation that

$$\mathcal{LA} = \{(\boldsymbol{x}, \boldsymbol{y}) | (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{A} \wedge f(\boldsymbol{x}) = f(\boldsymbol{y})\}$$

- Level accessibility induces a partitioning of the search space into level-accessible sets $\mathcal{S} = \bigcup_{i=1} \mathcal{V}_i$

- In ones-max there are $N + 1$ level accessible sets

Level Accessible

- We define accessibility as the relation

$$\mathcal{A} = \{(\boldsymbol{x}, \boldsymbol{y}) | \exists \pi(\boldsymbol{x}, \boldsymbol{y}) \wedge \forall \boldsymbol{x}_i \in \pi(\boldsymbol{x}, \boldsymbol{y}), f(\boldsymbol{x}_i) \leq f(\boldsymbol{x})\}$$

  i.e. there exists a path with no configuration exceeding the cost of the initial configuration

- Level accessibility is the equivalence relation that

$$\mathcal{LA} = \{(\boldsymbol{x}, \boldsymbol{y}) | (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{A} \wedge f(\boldsymbol{x}) = f(\boldsymbol{y})\}$$

- Level accessibility induces a partitioning of the search space into level-accessible sets $\mathcal{S} = \bigcup_{i=1} \mathcal{V}_i$

- In ones-max there are $N + 1$ level accessible sets

Level Accessible

- We define accessibility as the relation

$$\mathcal{A} = \{(\boldsymbol{x}, \boldsymbol{y}) | \exists \pi(\boldsymbol{x}, \boldsymbol{y}) \wedge \forall \boldsymbol{x}_i \in \pi(\boldsymbol{x}, \boldsymbol{y}), f(\boldsymbol{x}_i) \leq f(\boldsymbol{x})\}$$

  i.e. there exists a path with no configuration exceeding the cost of the initial configuration

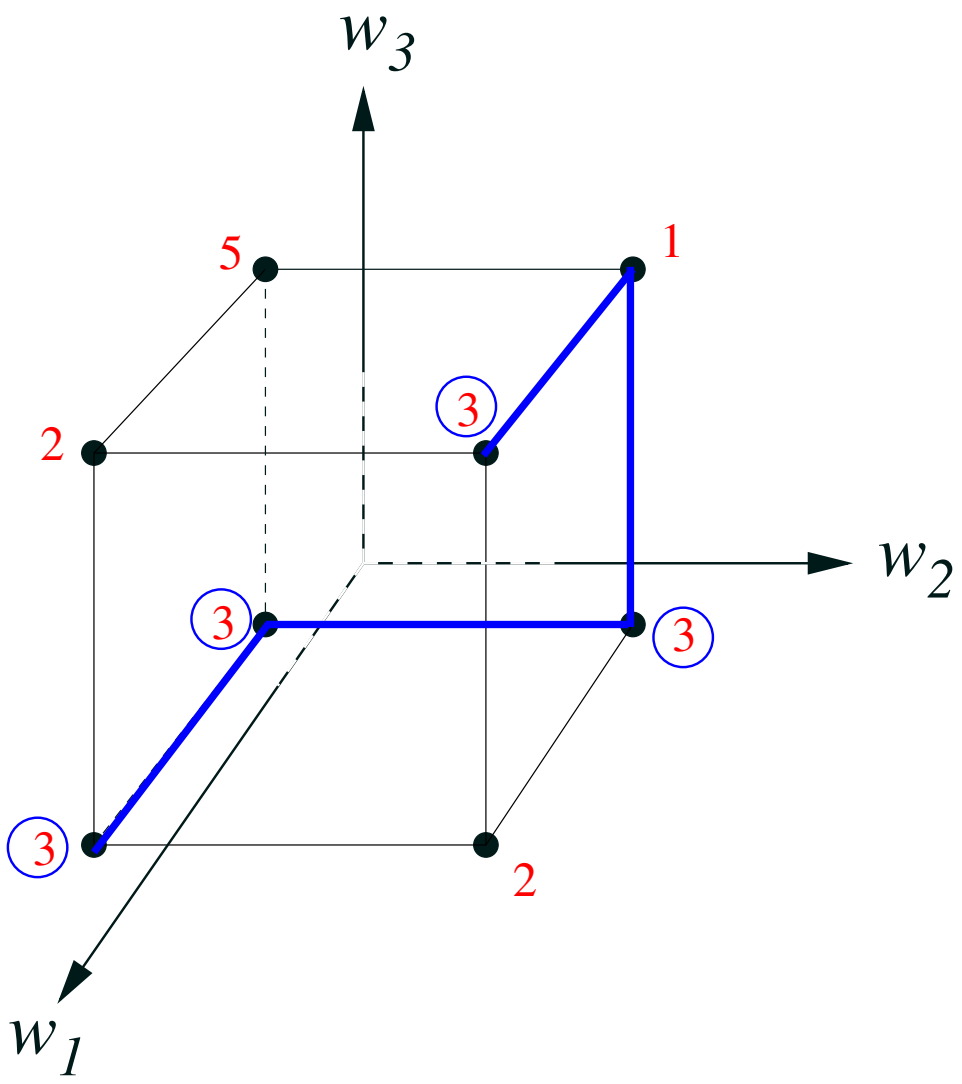
- Level accessibility is the equivalence relation that

$$\mathcal{LA} = \{(\boldsymbol{x}, \boldsymbol{y}) | (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{A} \wedge f(\boldsymbol{x}) = f(\boldsymbol{y})\}$$

- Level accessibility induces a partitioning of the search space into level-accessible sets $\mathcal{S} = \bigcup_{i=1} \mathcal{V}_i$

- In ones-max there are $N + 1$ level accessible sets

Level Accessible Sets



Barrier Trees

- We can define accessibility between level-accessible sets

$$\mathcal{A}_{LAS} = \{(\mathcal{V}_i, \mathcal{V}_j) \big| \exists \boldsymbol{x} \in \mathcal{V}_i, \exists \boldsymbol{y} \in \mathcal{V}_j, (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{A}\}$$

Barrier Trees

- We can define accessibility between level-accessible sets

$$\mathcal{A}_{LAS} = \{(\mathcal{V}_i, \mathcal{V}_j) \big| \exists \boldsymbol{x} \in \mathcal{V}_i, \exists \boldsymbol{y} \in \mathcal{V}_j, (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{A}\}$$

- Accessibility between level-accessible sets defines a partial ordering

Barrier Trees

- We can define accessibility between level-accessible sets

$$\mathcal{A}_{LAS} = \{(\mathcal{V}_i, \mathcal{V}_j) \big| \exists \boldsymbol{x} \in \mathcal{V}_i, \exists \boldsymbol{y} \in \mathcal{V}_j, (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{A}\}$$

- Accessibility between level-accessible sets defines a partial ordering

- The Barrier tree is the diagram of the partial ordering

Barrier Trees

- We can define accessibility between level-accessible sets

$$\mathcal{A}_{LAS} = \{(\mathcal{V}_i, \mathcal{V}_j) \big| \exists \boldsymbol{x} \in \mathcal{V}_i, \exists \boldsymbol{y} \in \mathcal{V}_j, (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{A}\}$$

- Accessibility between level-accessible sets defines a partial ordering

- The Barrier tree is the diagram of the partial ordering i.e. graph of direct predecessor

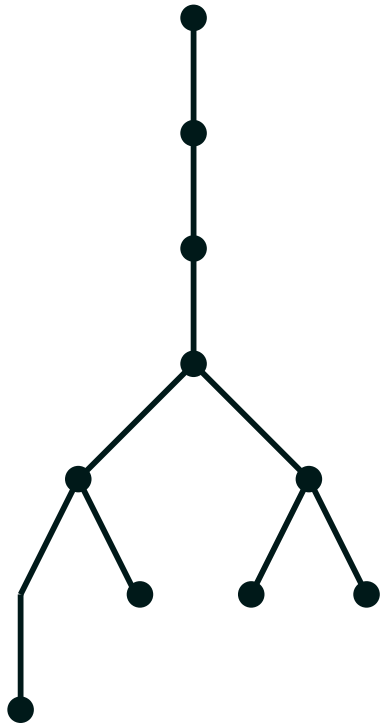
Barrier Trees

- We can define accessibility between level-accessible sets

$$\mathcal{A}_{LAS} = \{(\mathcal{V}_i, \mathcal{V}_j) \big| \exists \boldsymbol{x} \in \mathcal{V}_i, \exists \boldsymbol{y} \in \mathcal{V}_j, (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{A}\}$$

- Accessibility between level-accessible sets defines a partial ordering

- The Barrier tree is the diagram of the partial ordering i.e. graph of direct predecessor



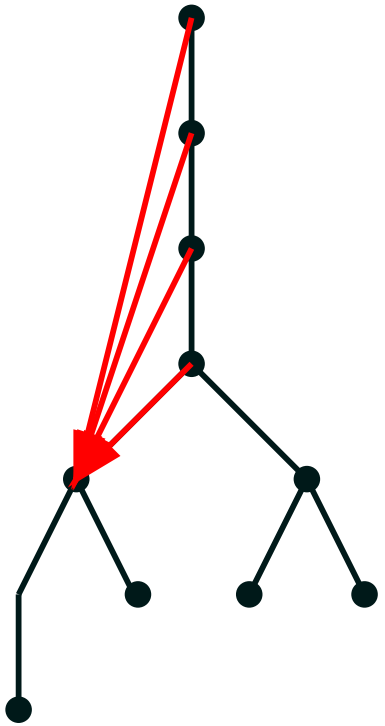
Barrier Trees

- We can define accessibility between level-accessible sets

$$\mathcal{A}_{LAS} = \{(\mathcal{V}_i, \mathcal{V}_j) \big| \exists \boldsymbol{x} \in \mathcal{V}_i, \exists \boldsymbol{y} \in \mathcal{V}_j, (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{A}\}$$

- Accessibility between level-accessible sets defines a partial ordering

- The Barrier tree is the diagram of the partial ordering i.e. graph of direct predecessor



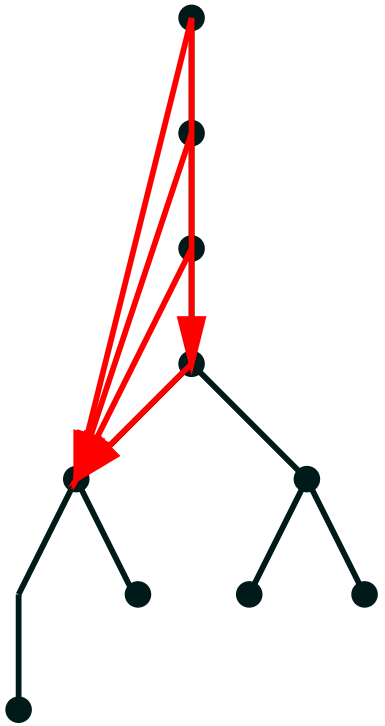
Barrier Trees

- We can define accessibility between level-accessible sets

$$\mathcal{A}_{LAS} = \{(\mathcal{V}_i, \mathcal{V}_j) \big| \exists \boldsymbol{x} \in \mathcal{V}_i, \exists \boldsymbol{y} \in \mathcal{V}_j, (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{A}\}$$

- Accessibility between level-accessible sets defines a partial ordering

- The Barrier tree is the diagram of the partial ordering i.e. graph of direct predecessor



Computing Barrier Trees

- Finding the level-accessible sets and Barrier trees can be computed efficiently, $O(|\mathcal{S}| \times |\mathcal{N}|)$, using a flooding algorithm
- For larger problems we can compute the low-cost part of the Barrier tree using a modified branch and bound algorithm
- The statistical properties of the rest of the tree can be estimated using sampling techniques

Computing Barrier Trees

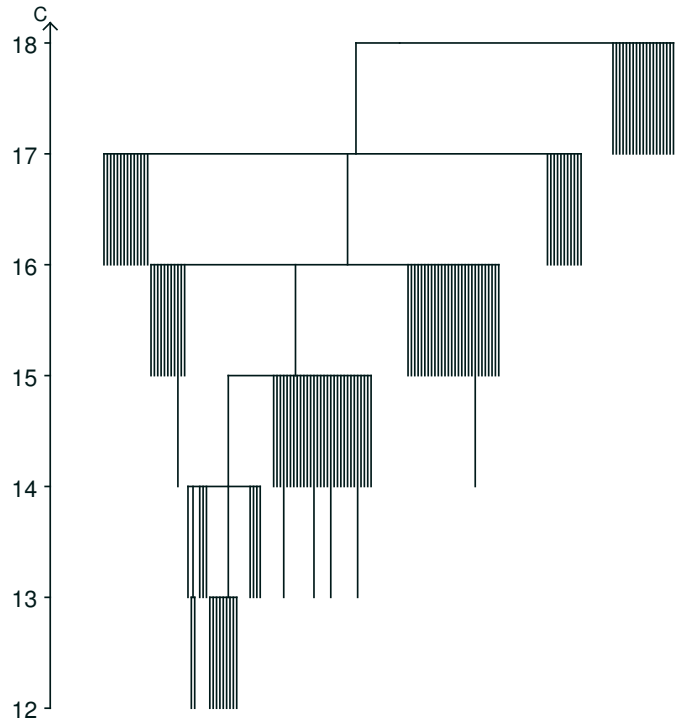
- Finding the level-accessible sets and Barrier trees can be computed efficiently, $O(|\mathcal{S}| \times |\mathcal{N}|)$, using a flooding algorithm
- For larger problems we can compute the low-cost part of the Barrier tree using a modified branch and bound algorithm
- The statistical properties of the rest of the tree can be estimated using sampling techniques

Computing Barrier Trees

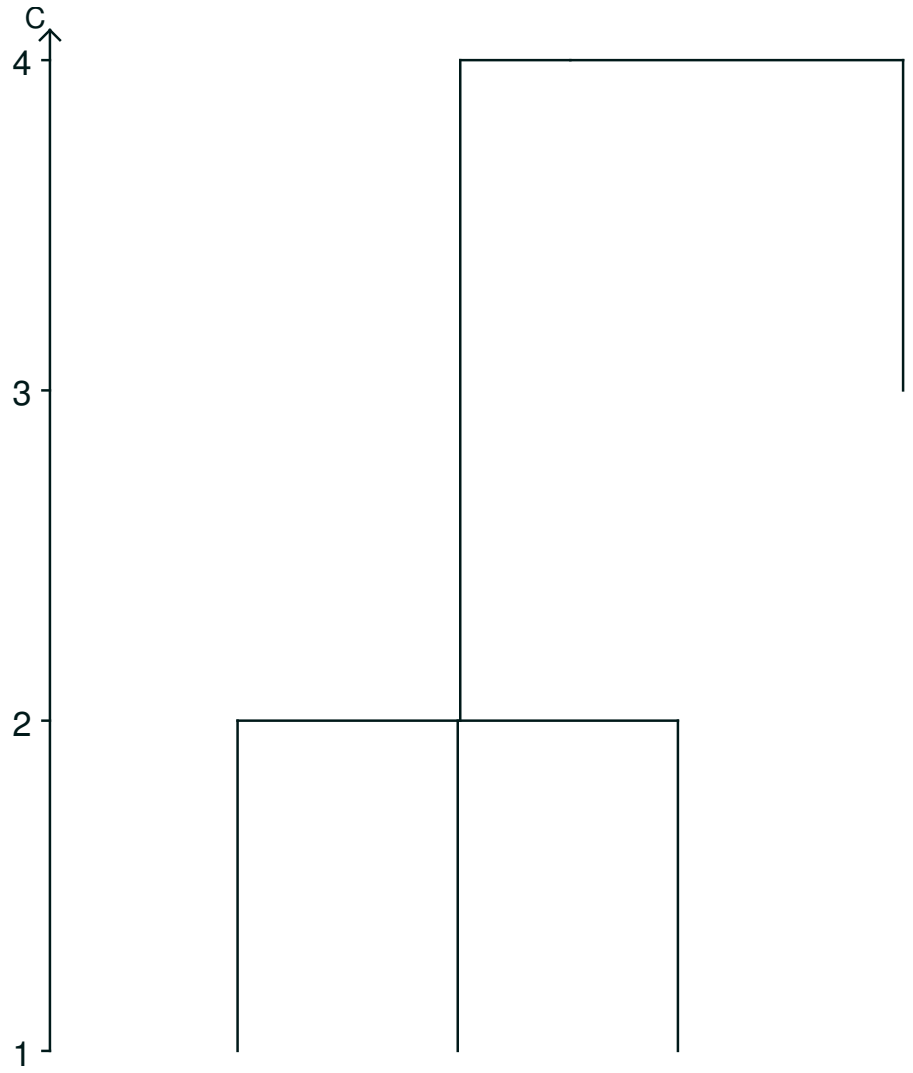
- Finding the level-accessible sets and Barrier trees can be computed efficiently, $O(|\mathcal{S}| \times |\mathcal{N}|)$, using a flooding algorithm
- For larger problems we can compute the low-cost part of the Barrier tree using a modified branch and bound algorithm
- The statistical properties of the rest of the tree can be estimated using sampling techniques

Outline

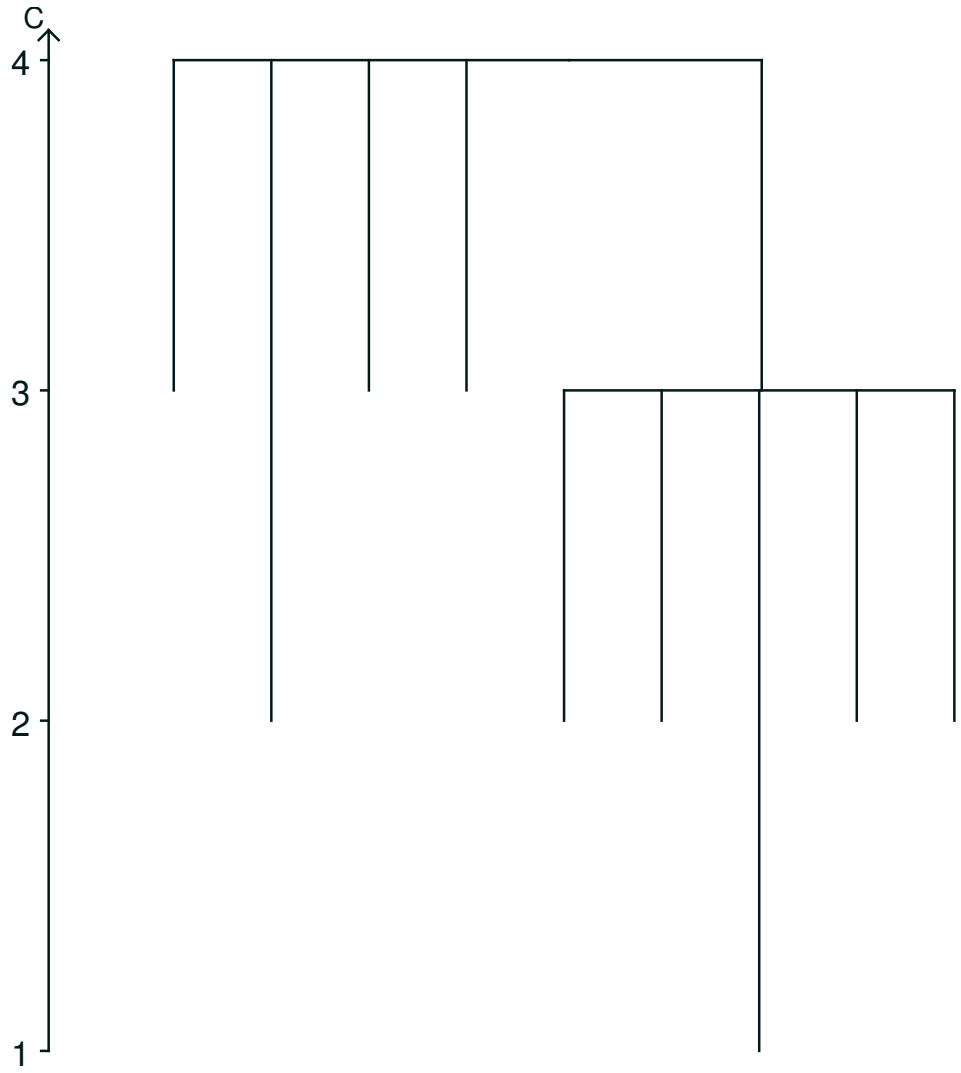
1. Optimisation Problems
2. Cost Landscapes
3. Barrier Trees
4. Example Instances
5. Mapping Configurations
6. Modelling the Problem



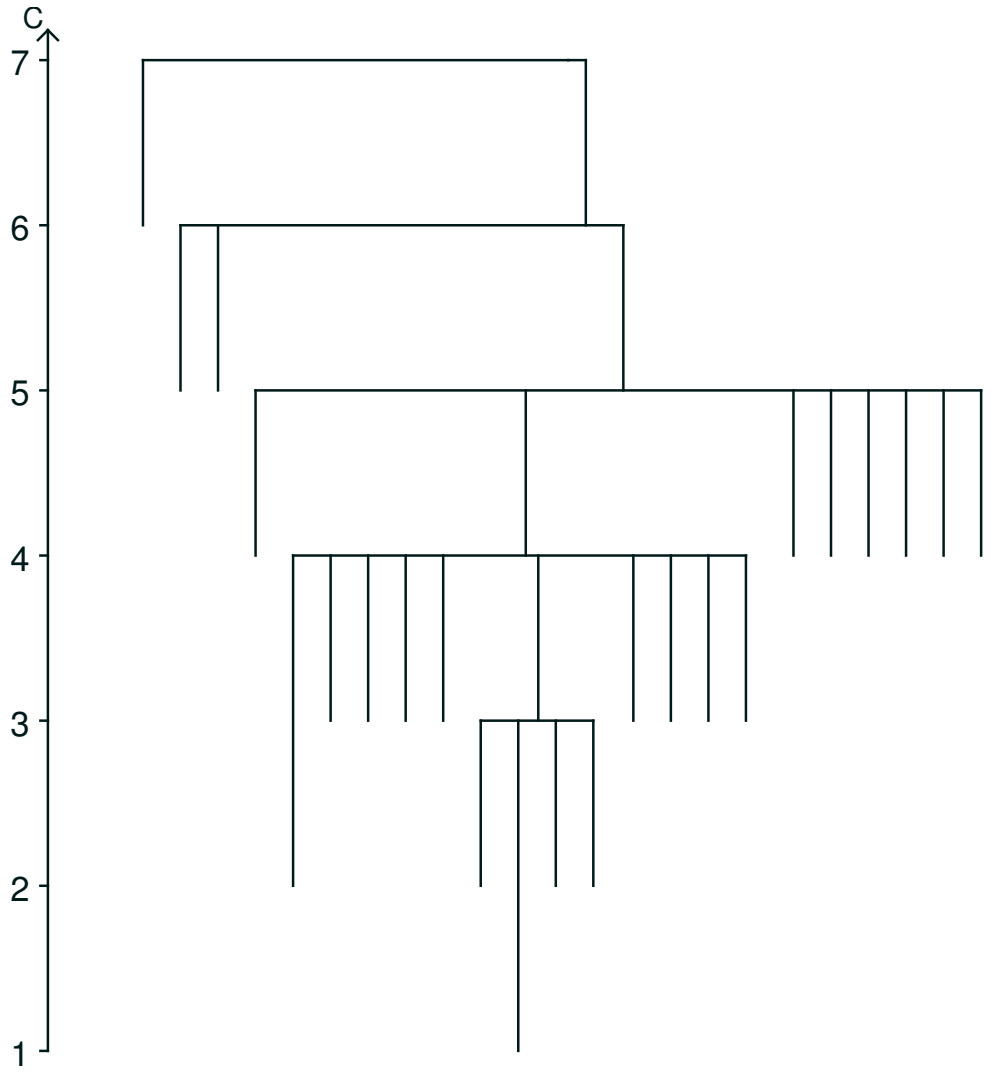
MAX-3-SAT $\alpha = M/N = 5$ $N = 10$



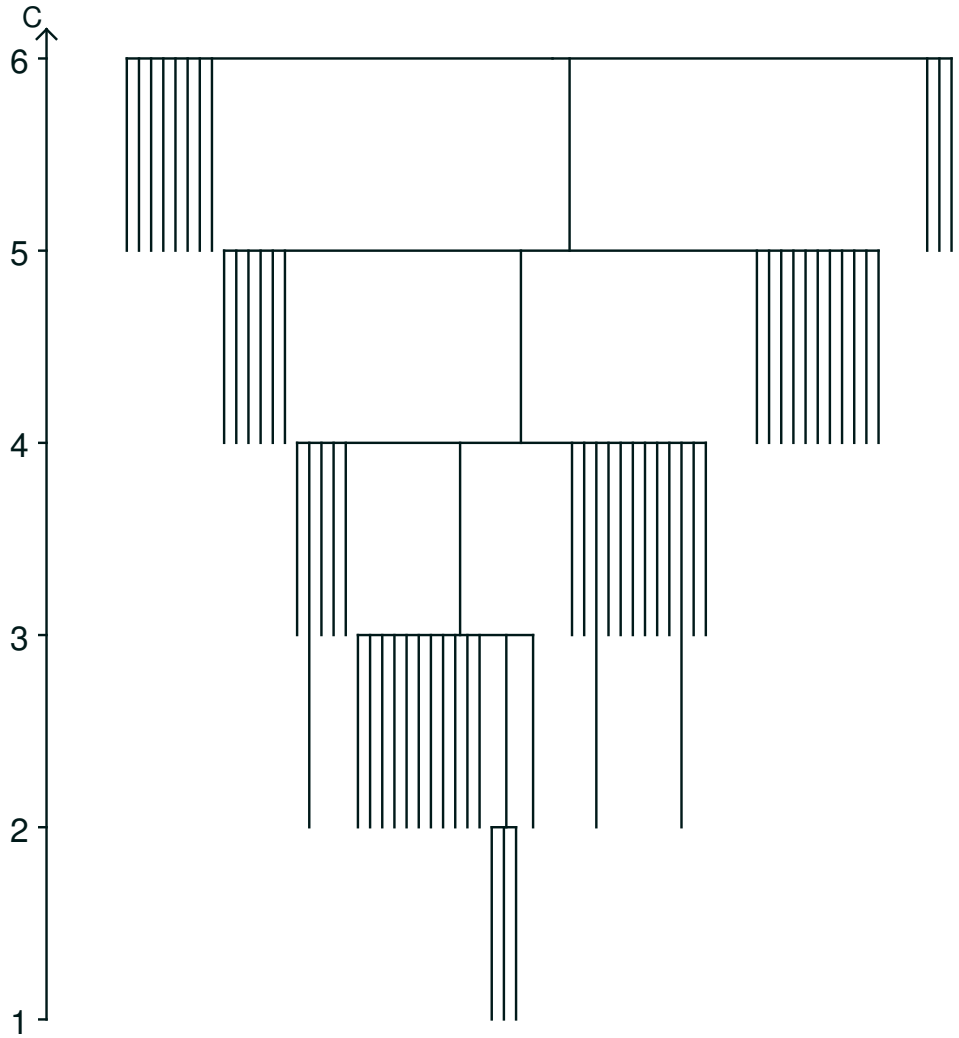
MAX-3-SAT $\alpha = M/N = 5$ $N = 20$



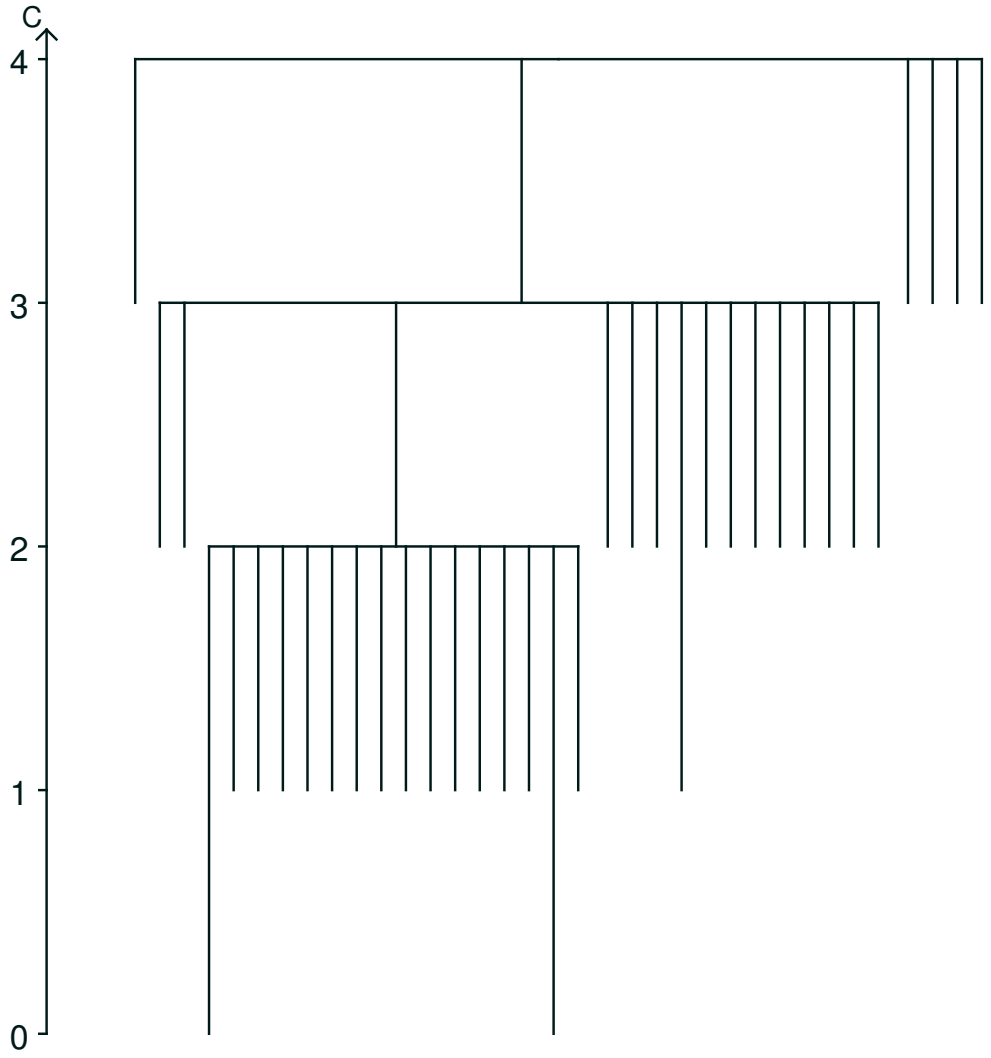
MAX-3-SAT $\alpha = M/N = 5$ $N = 30$



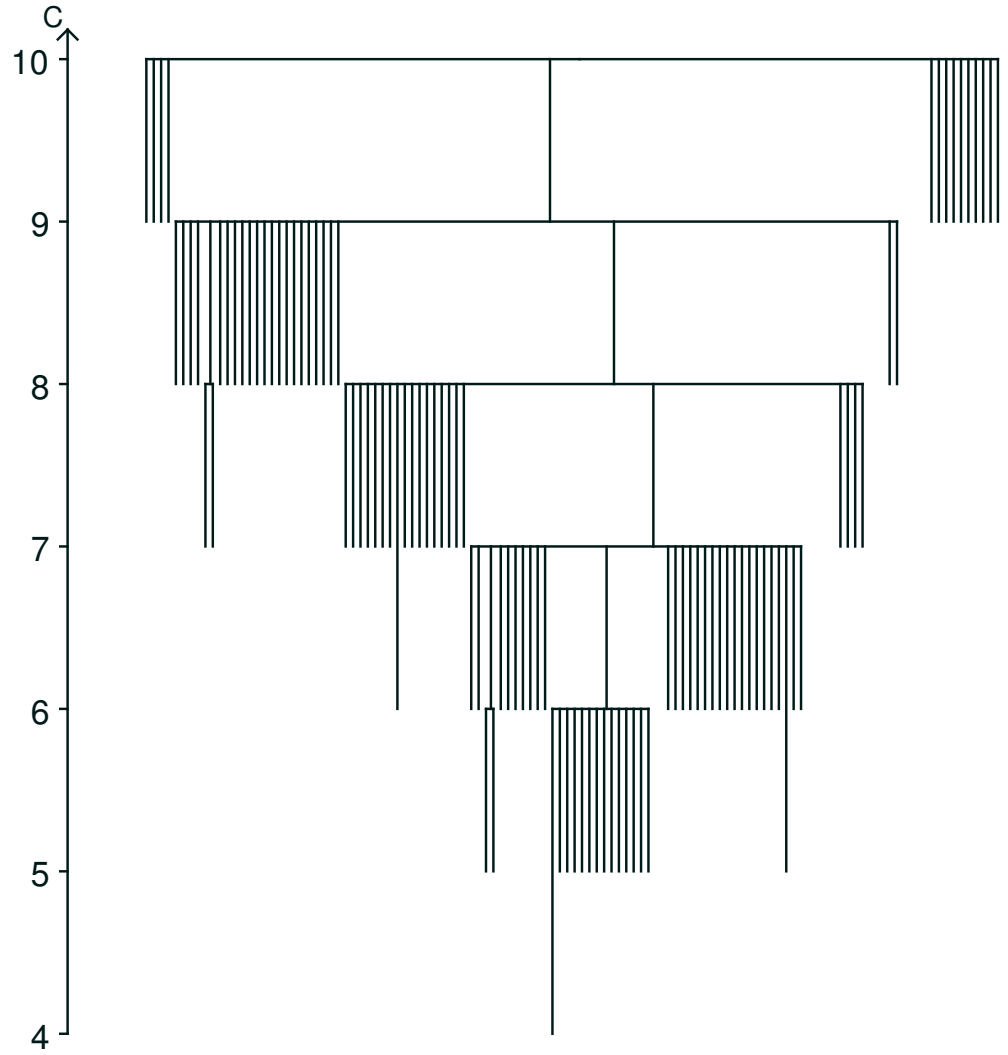
MAX-3-SAT $\alpha = M/N = 5$ $N = 40$



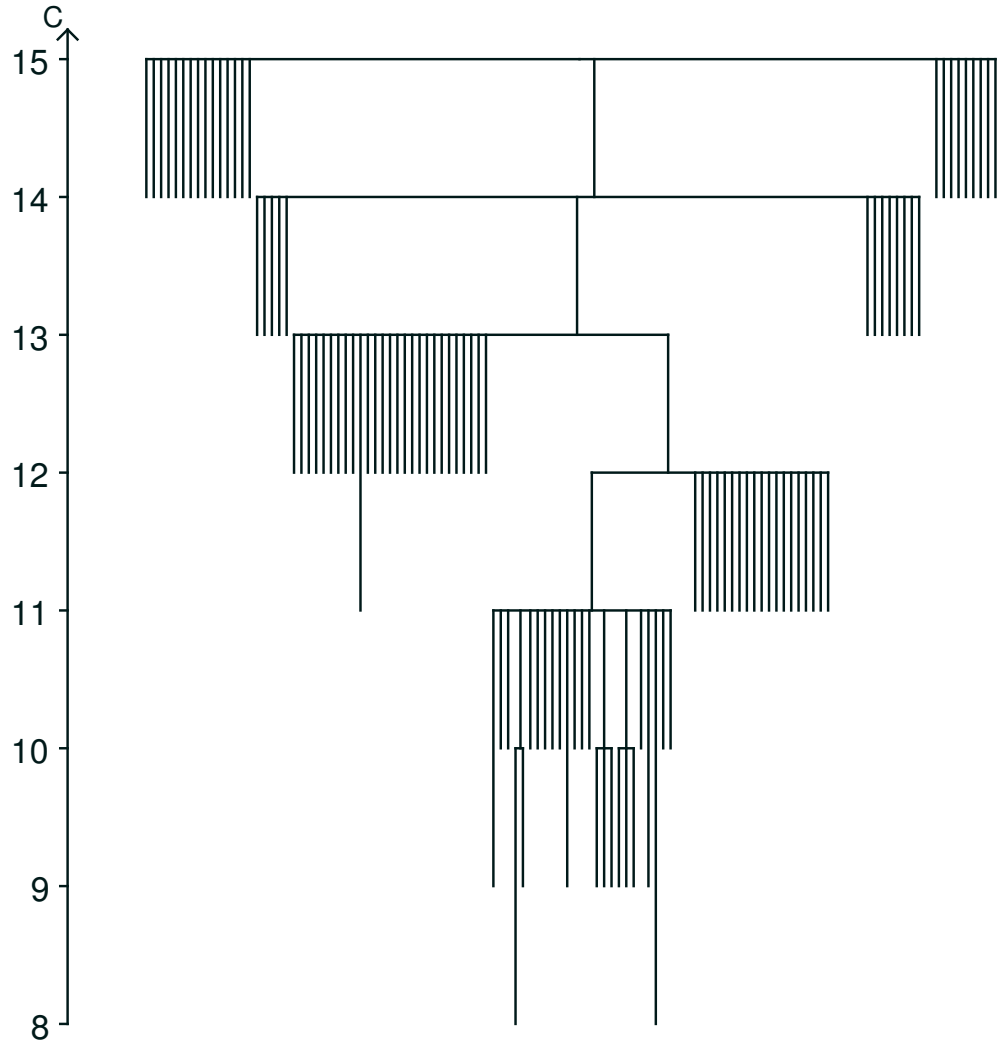
MAX-3-SAT $N = 40$ $\alpha = M/N = 4$



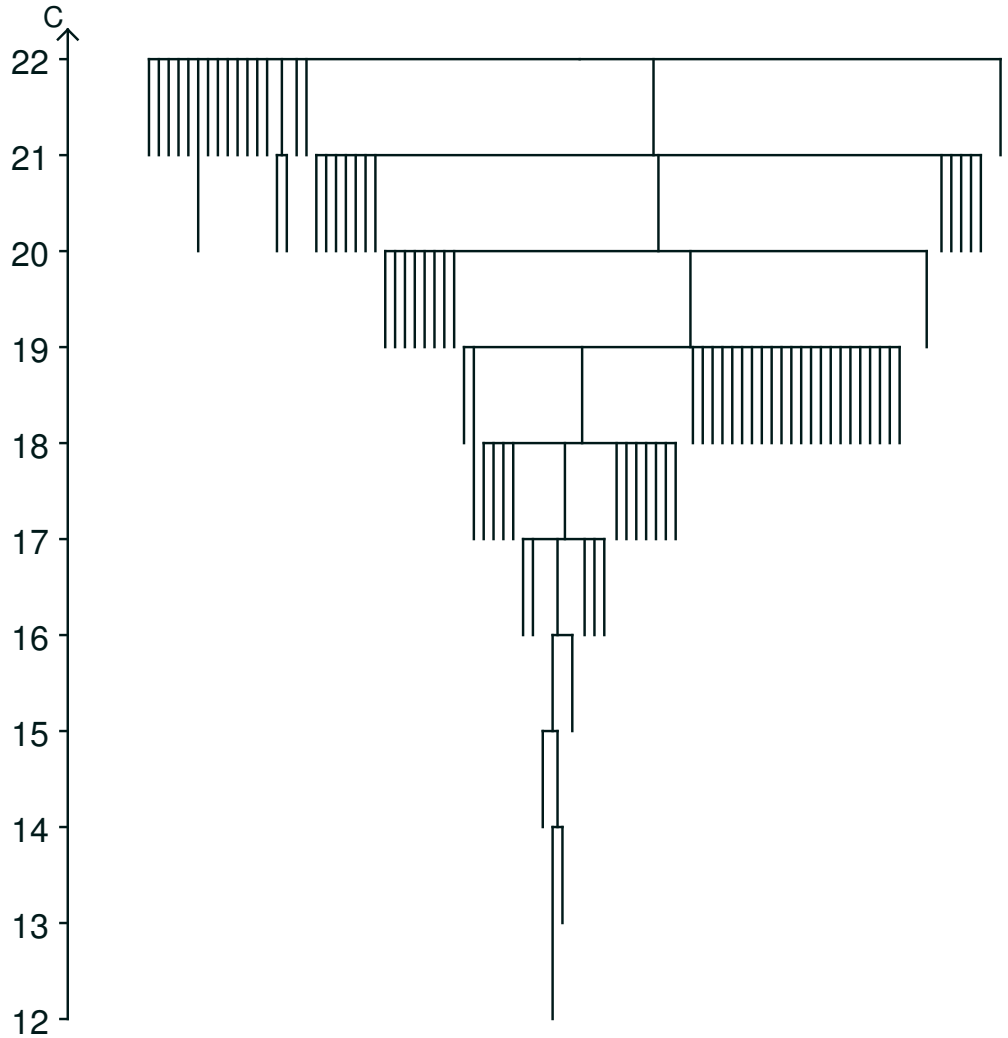
MAX-3-SAT $N = 40$ $\alpha = M/N = 6$



MAX-3-SAT $N = 40$ $\alpha = M/N = 8$

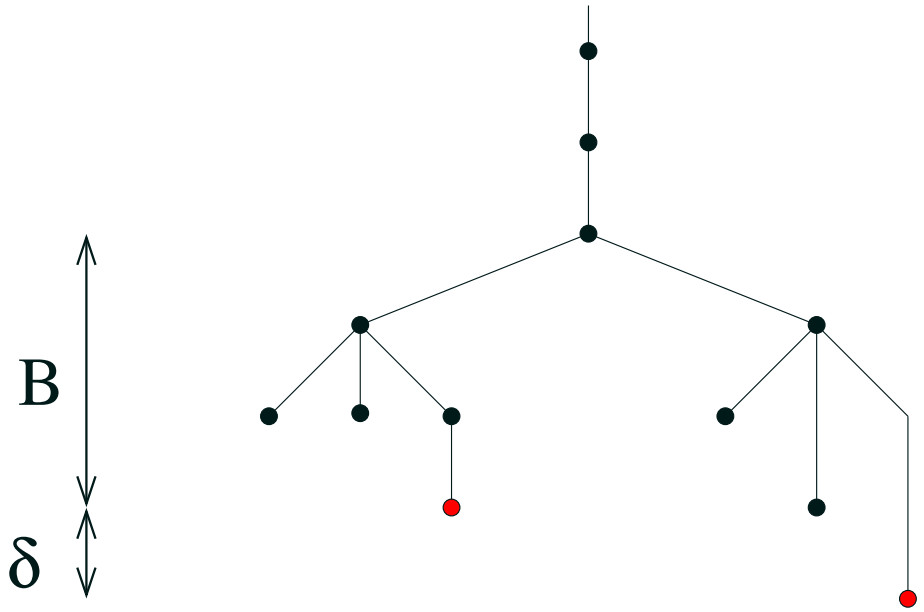


MAX-3-SAT $N = 40$ $\alpha = M/N = 10$



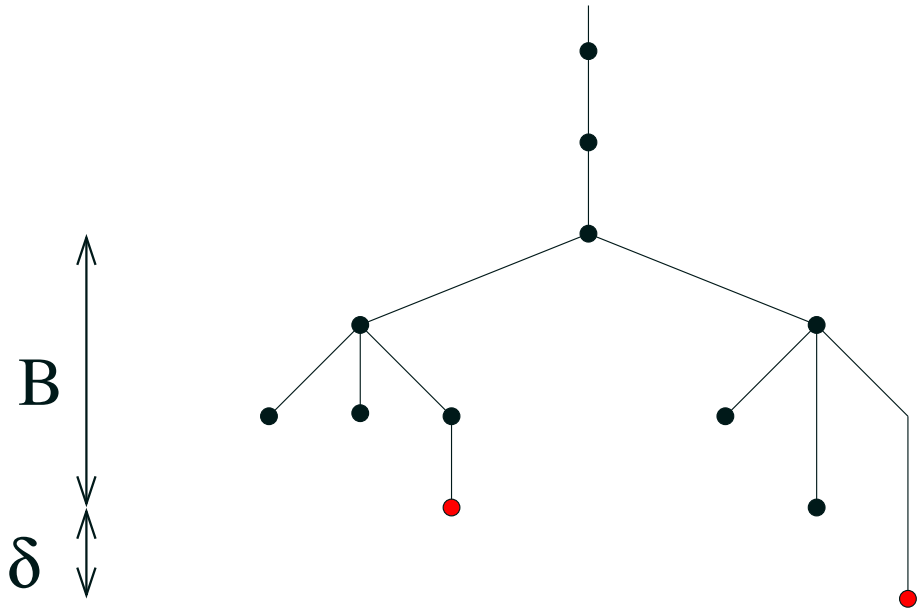
Phase Transition in Number Partitioning

- Are phase transitions evident in barrier trees?
- Studied by Stadler, Hordijk and Fontanari
- Very little evidence in shape
- Phase transition in difficulty B/δ



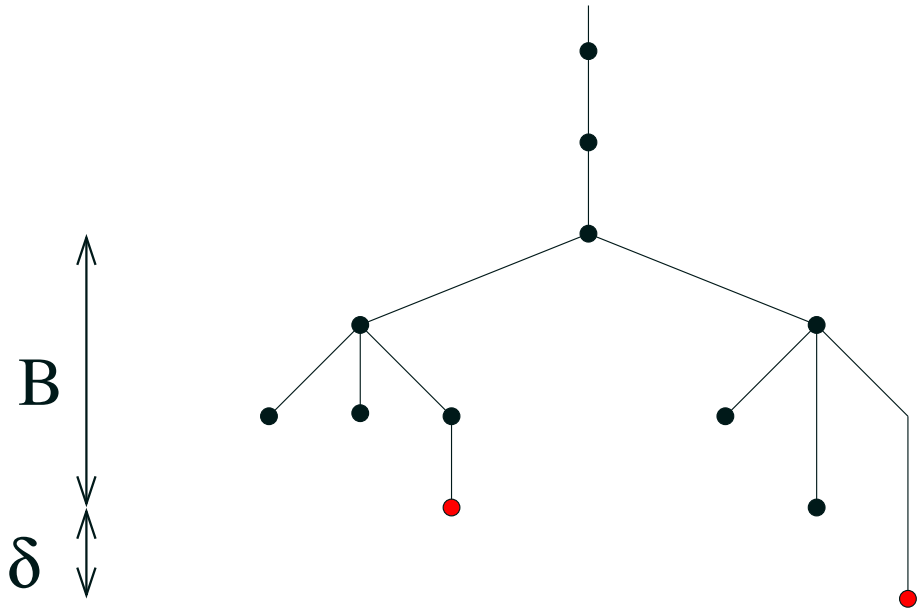
Phase Transition in Number Partitioning

- Are phase transitions evident in barrier trees?
- Studied by Stadler, Hordijk and Fontanari
- Very little evidence in shape
- Phase transition in difficulty B/δ



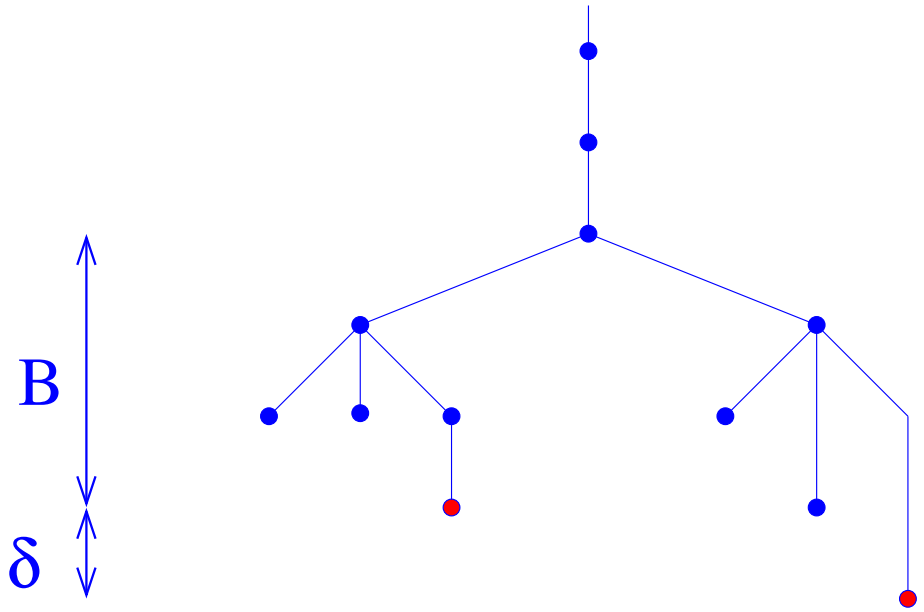
Phase Transition in Number Partitioning

- Are phase transitions evident in barrier trees?
- Studied by Stadler, Hordijk and Fontanari
- Very little evidence in shape
- Phase transition in difficulty B/δ



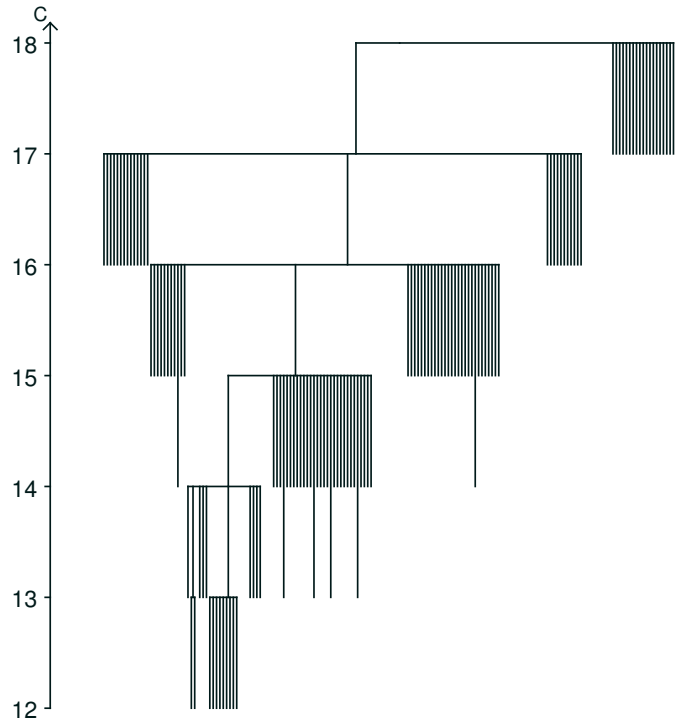
Phase Transition in Number Partitioning

- Are phase transitions evident in barrier trees?
- Studied by Stadler, Hordijk and Fontanari
- Very little evidence in shape
- Phase transition in difficulty B/δ

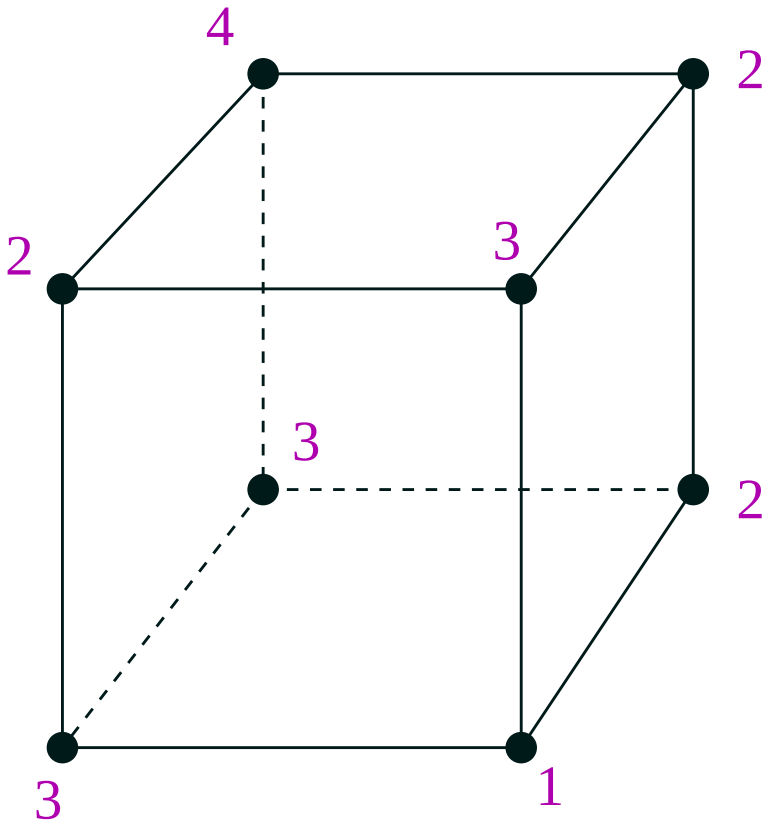


Outline

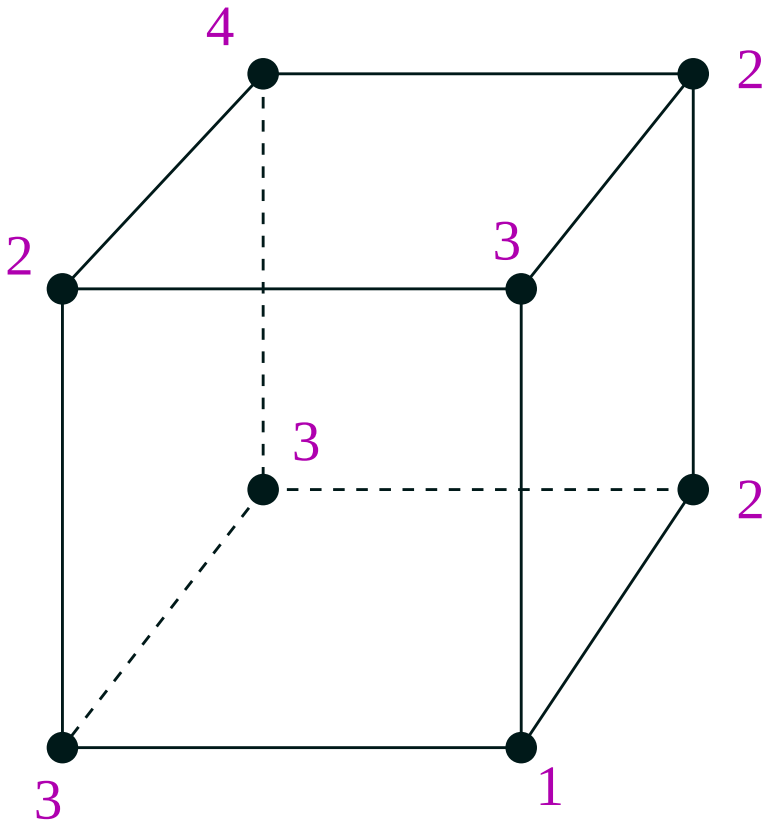
1. Optimisation Problems
2. Cost Landscapes
3. Barrier Trees
4. Example Instances
5. Mapping Configurations
6. Modelling the Problem

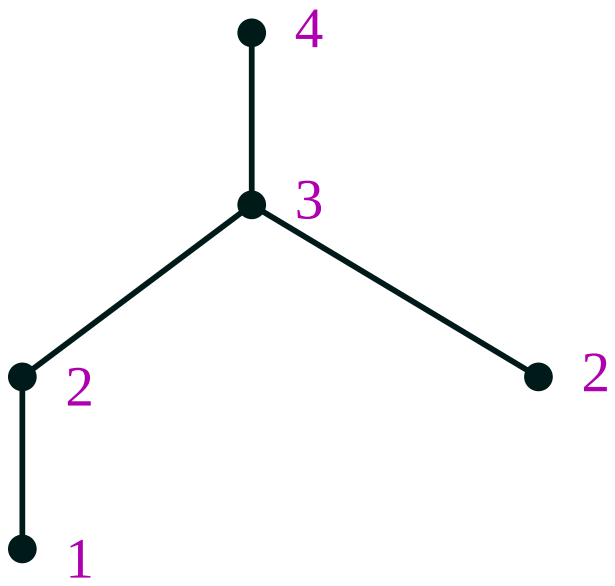


Mappings Configurations

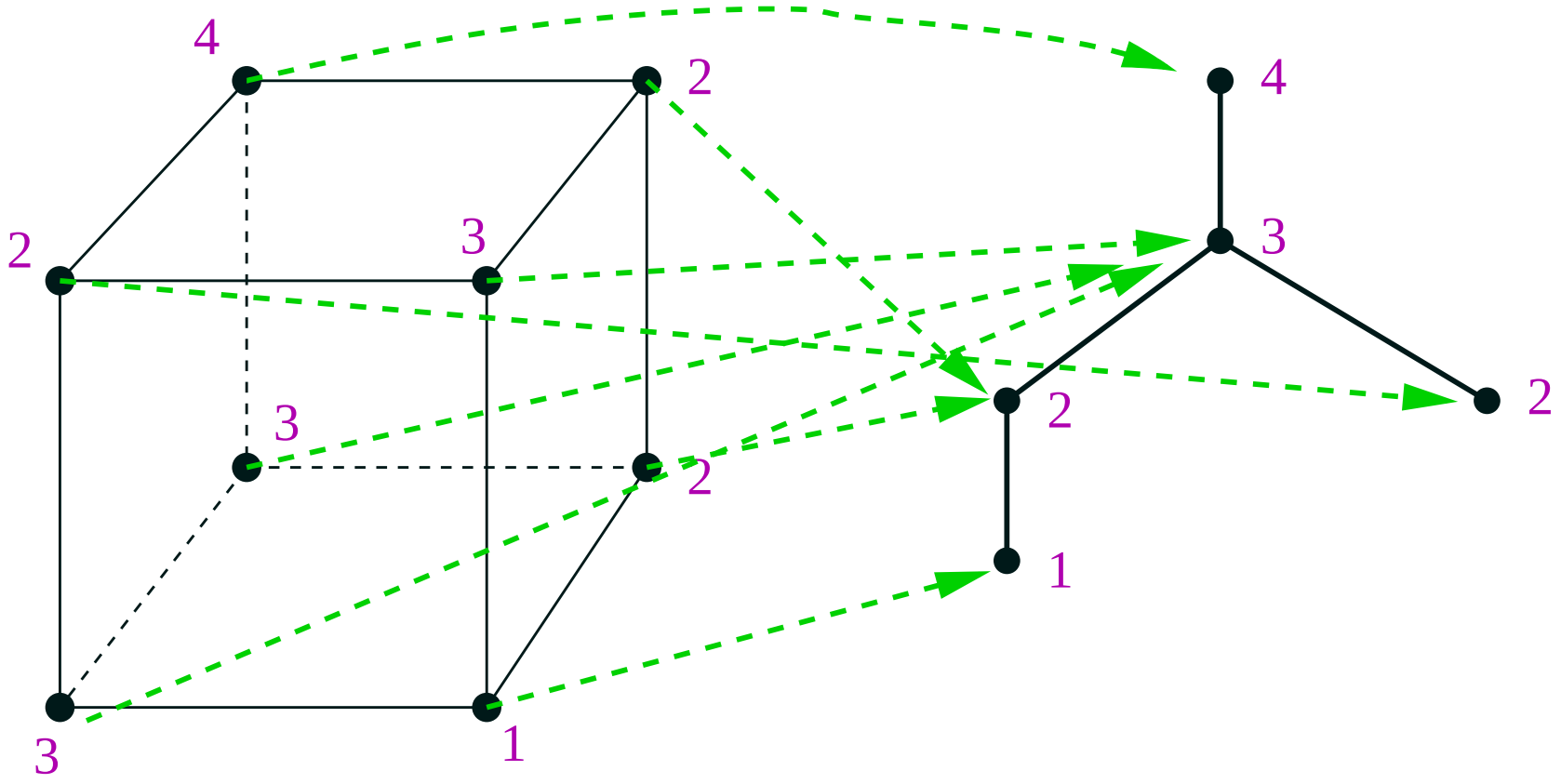


Mappings Configurations





Mappings Configurations



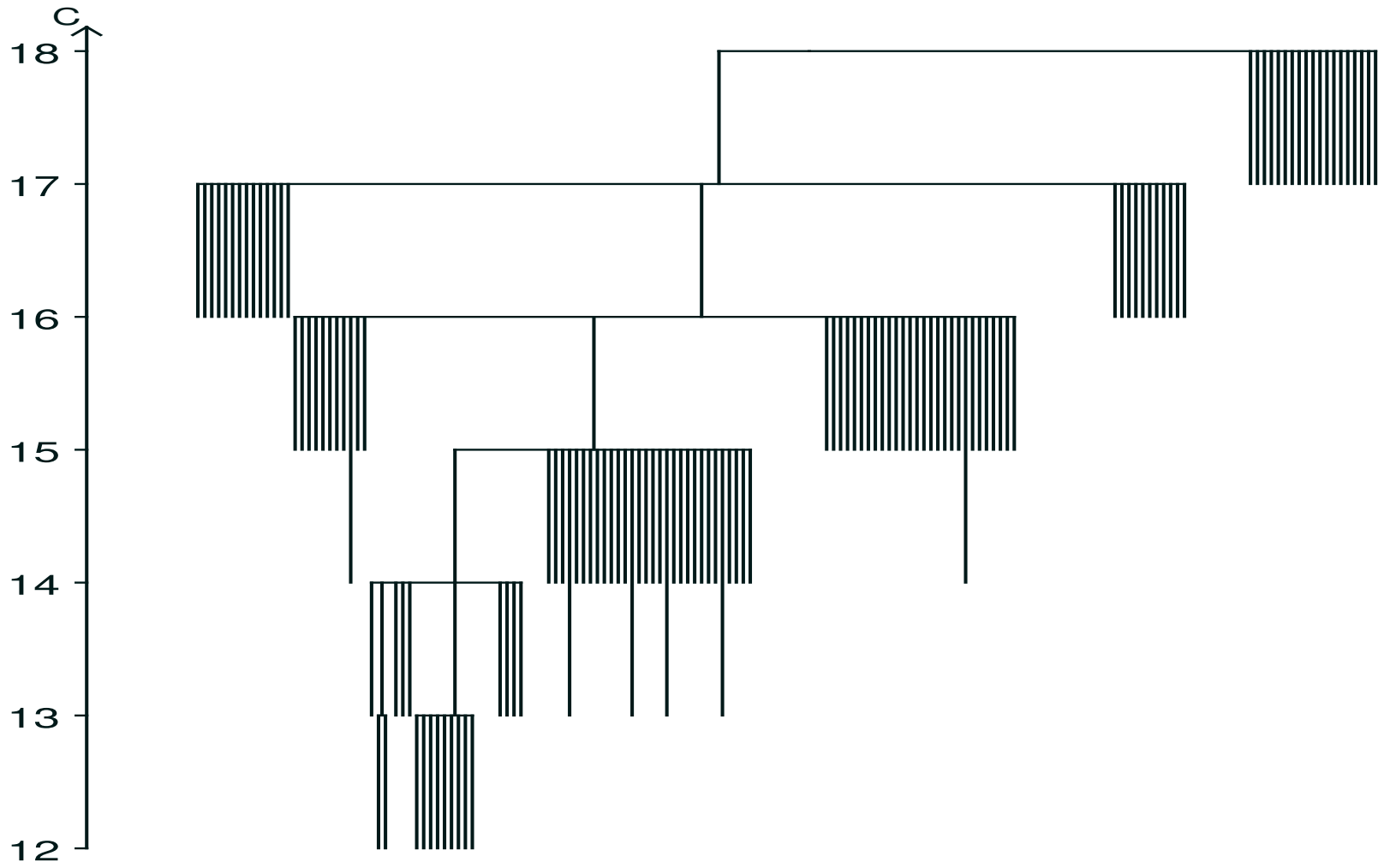
Mapping Configurations

- Every configuration is mapped to a node of the Barrier tree
- We can study statistical properties of the tree vertices
  - ⋆ number of configurations in a state
  - ⋆ Stringiness of local minima
  - ⋆ Distance to global solution

Mapping Configurations

- Every configuration is mapped to a node of the Barrier tree
- We can study statistical properties of the tree vertices
  - ⋆ number of configurations in a state
  - ⋆ Stringiness of local minima
  - ⋆ Distance to global solution

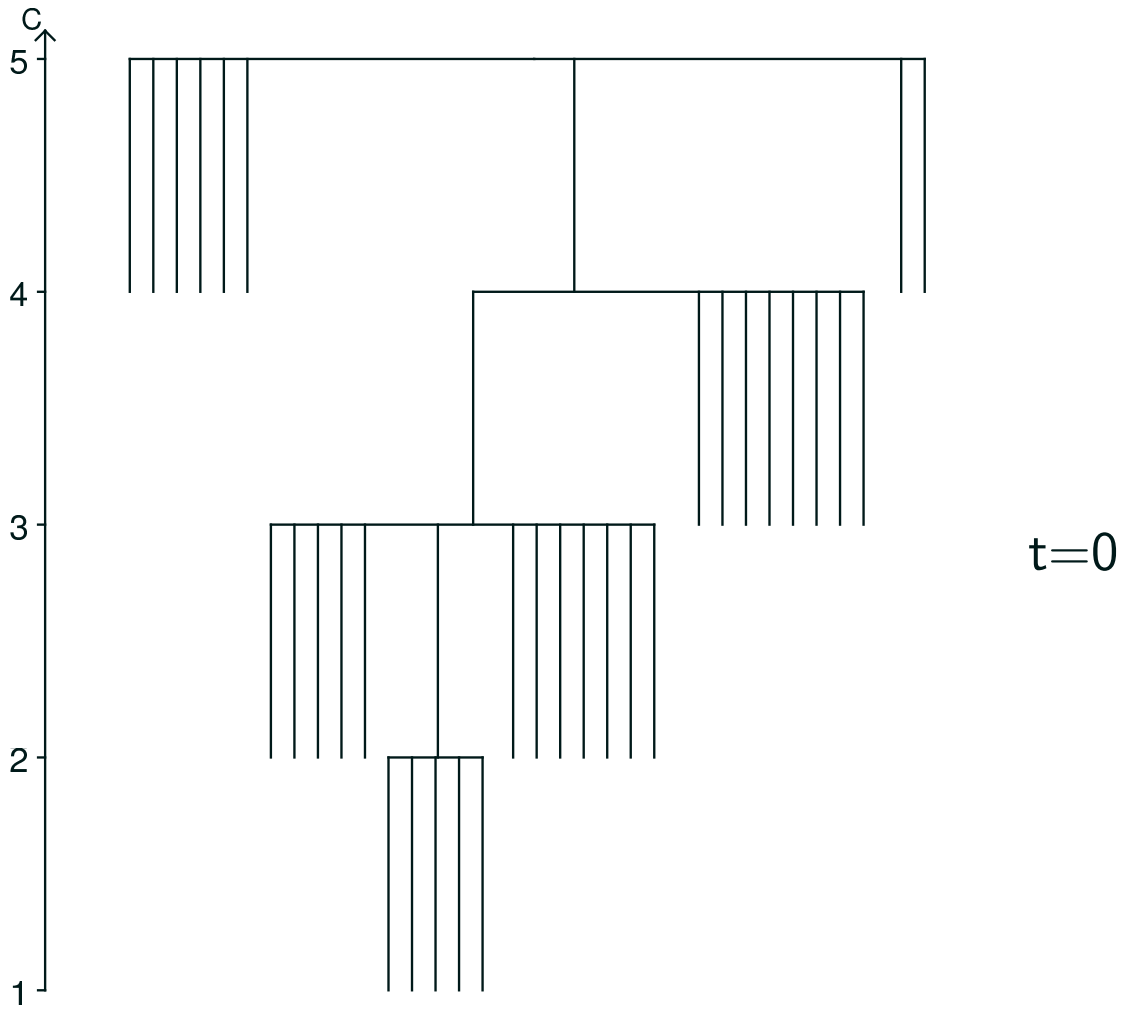
Max-3-Sat $N = 40$ $\alpha = M/N = 8$



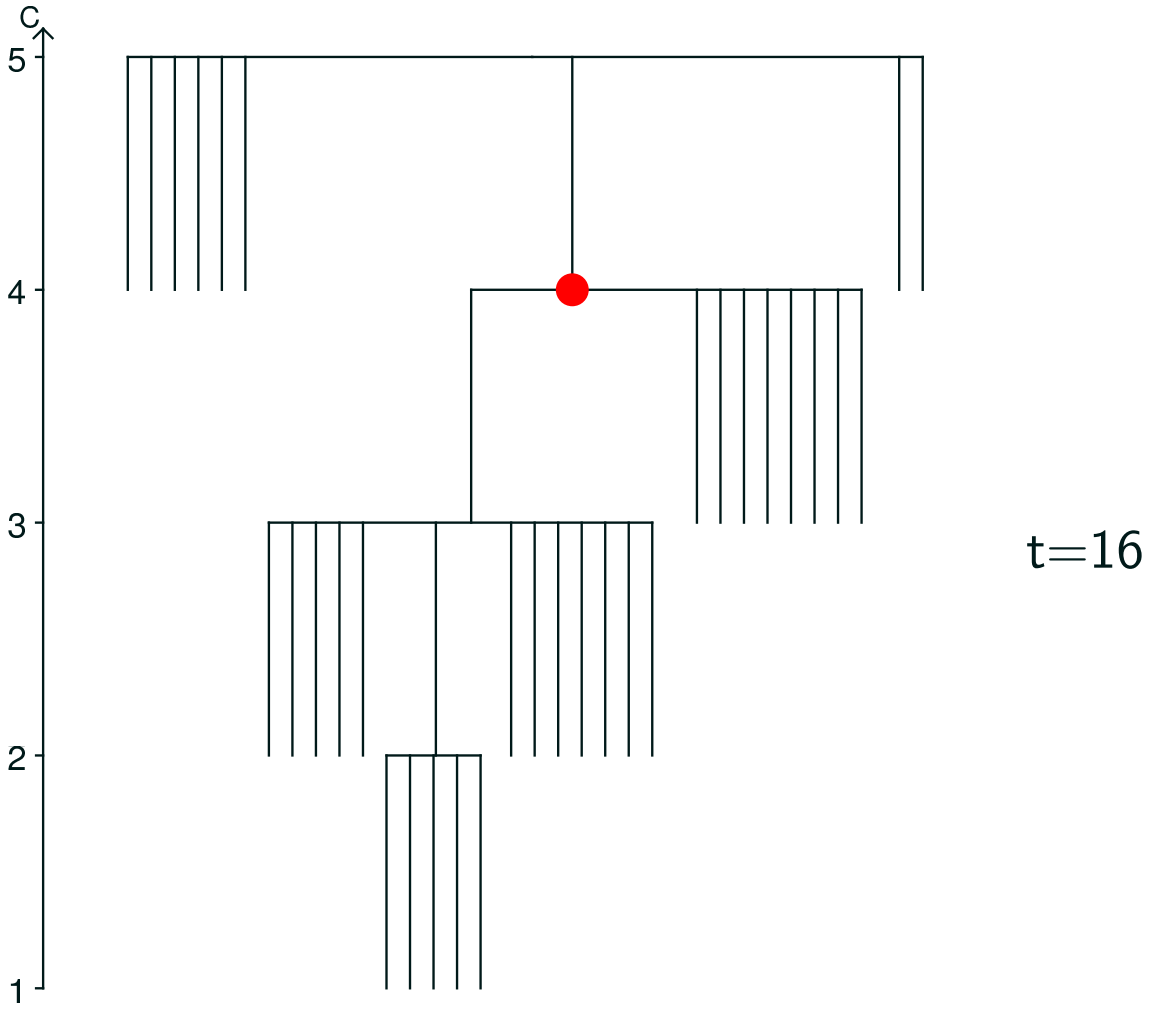
Statistics

Cost	# of configs	# of minima	Mean basin size	# minima depth > 1	% configs local minima	# of saddle points	% descents runs
16	210560	25	2.44	0	0.029	1	2.06
15	46518	37	4.29	0	0.34	1	9.09
14	8336	28	3.93	2	1.3	1	18.04
13	1041	12	10.92	4	12.6	2	18.45
12	47	11	4.27	0	100.0	0	50.71

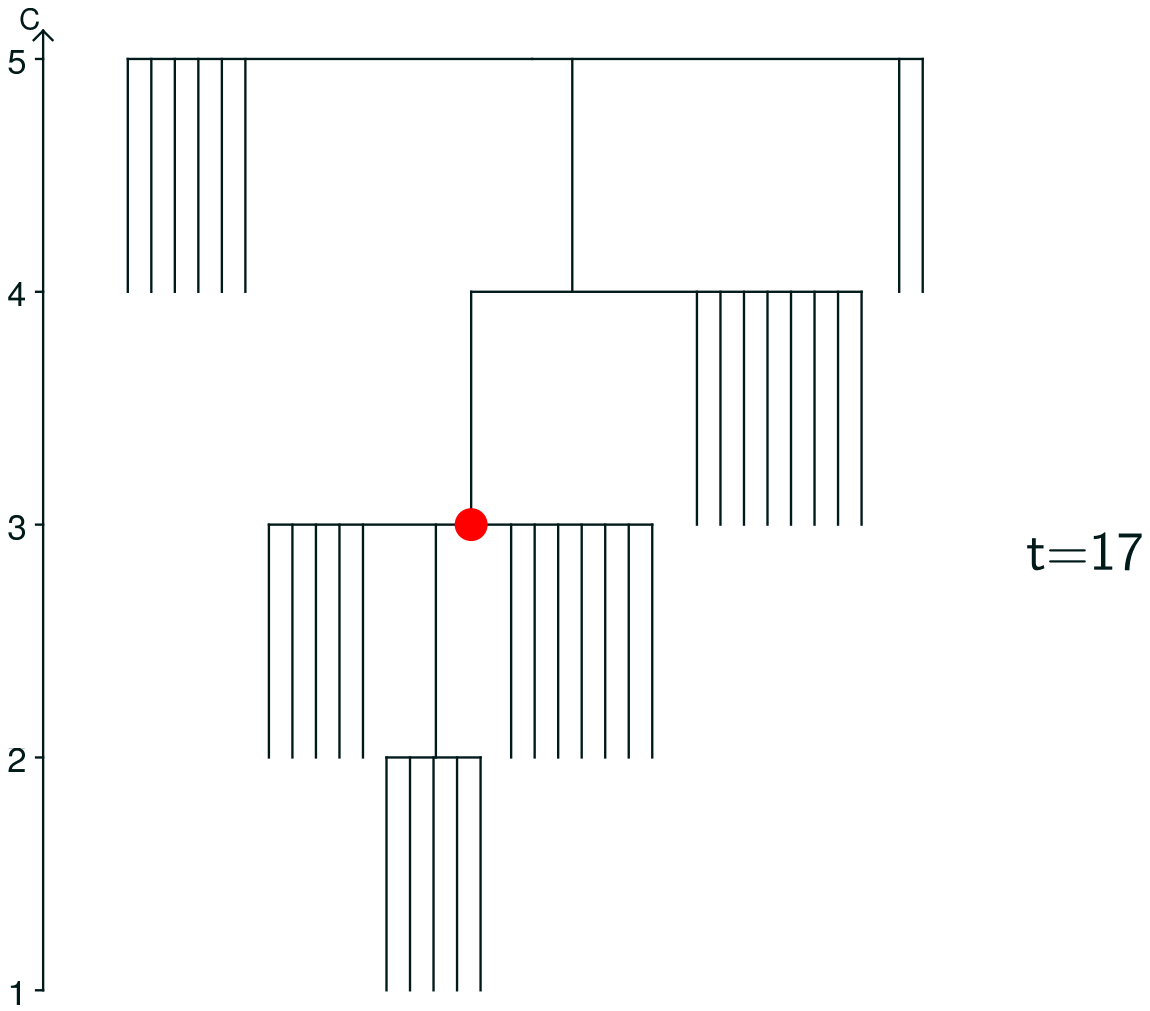
Simulated Annealing 40 Variables



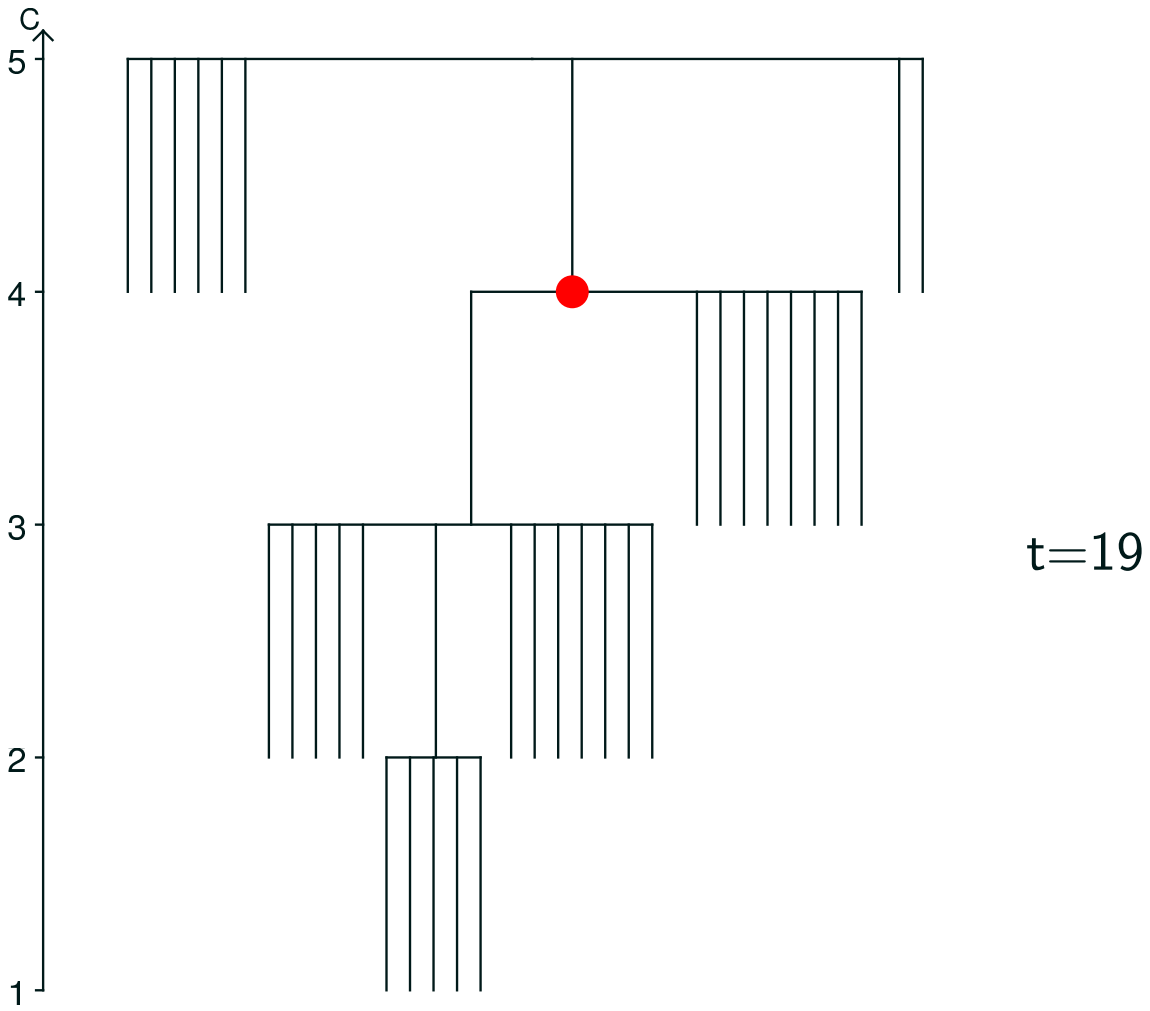
Simulated Annealing 40 Variables



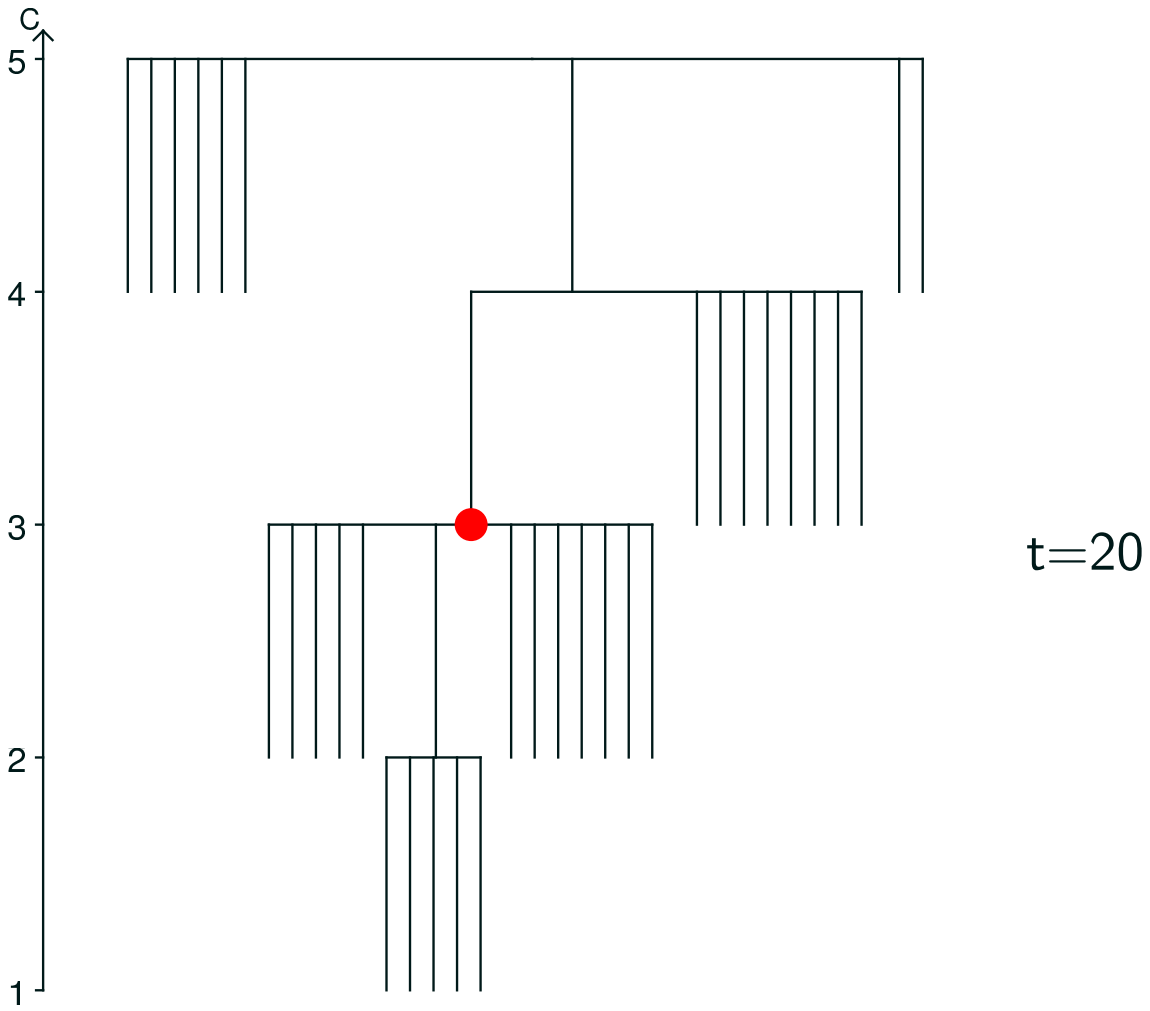
Simulated Annealing 40 Variables



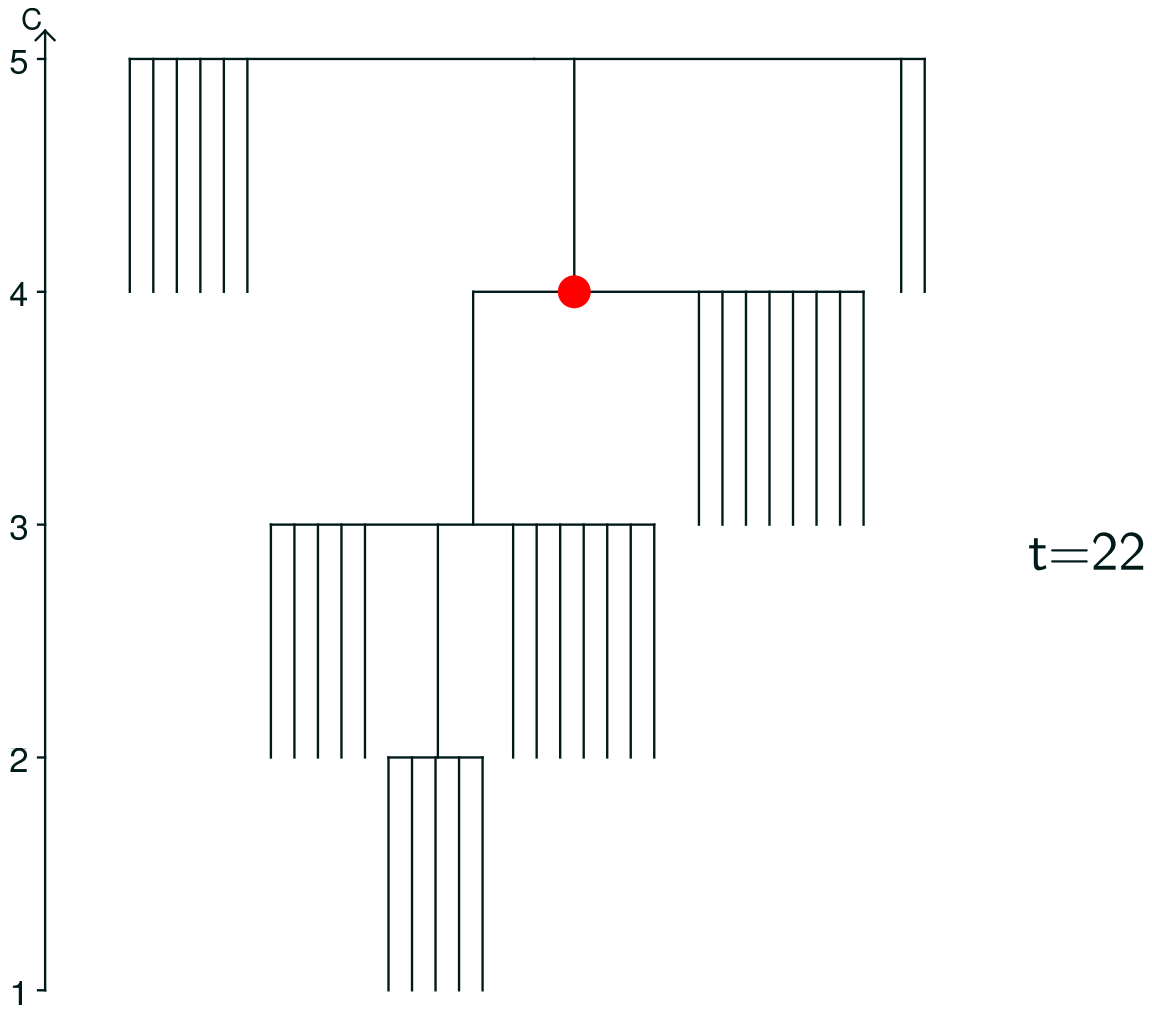
Simulated Annealing 40 Variables



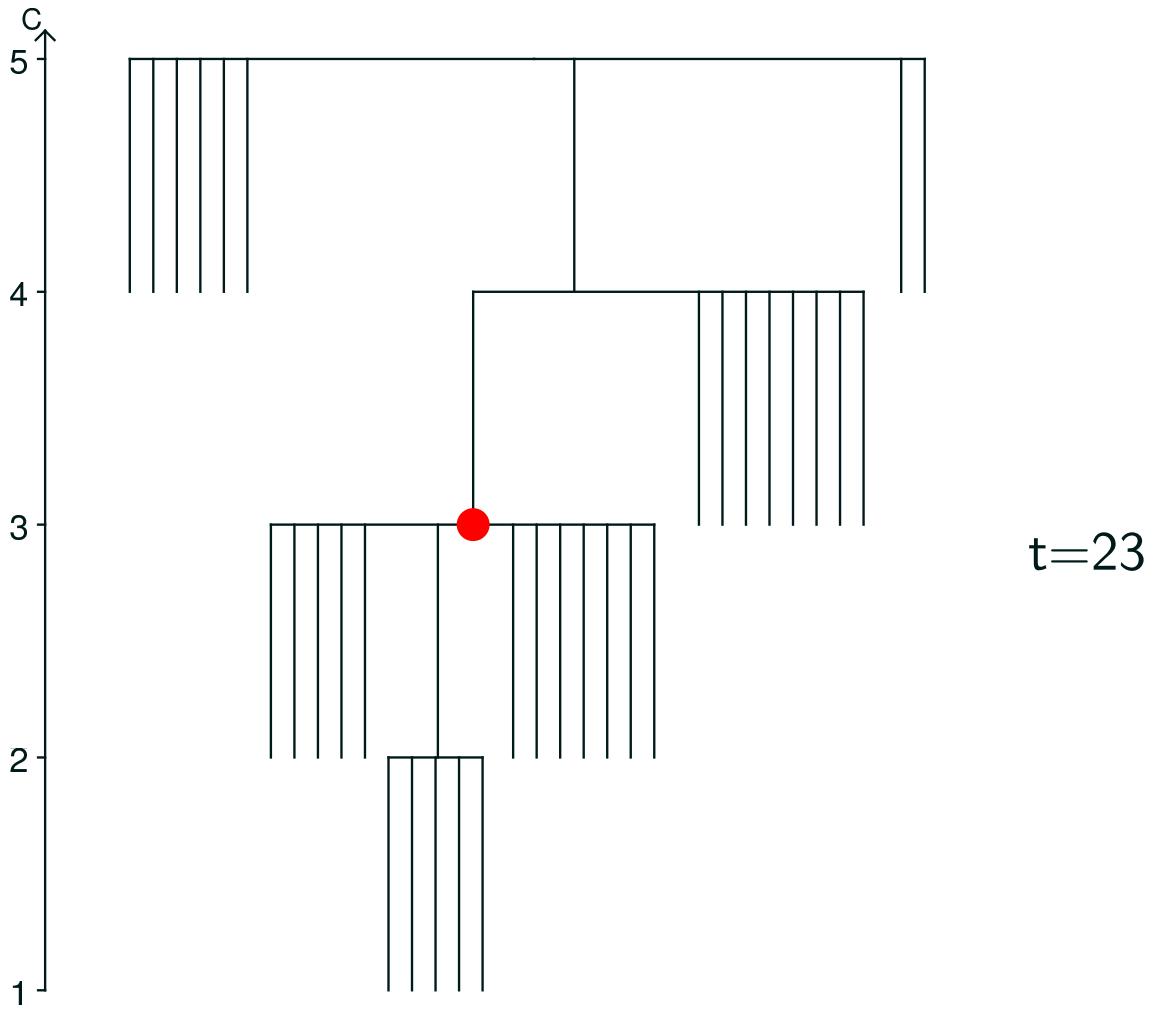
Simulated Annealing 40 Variables



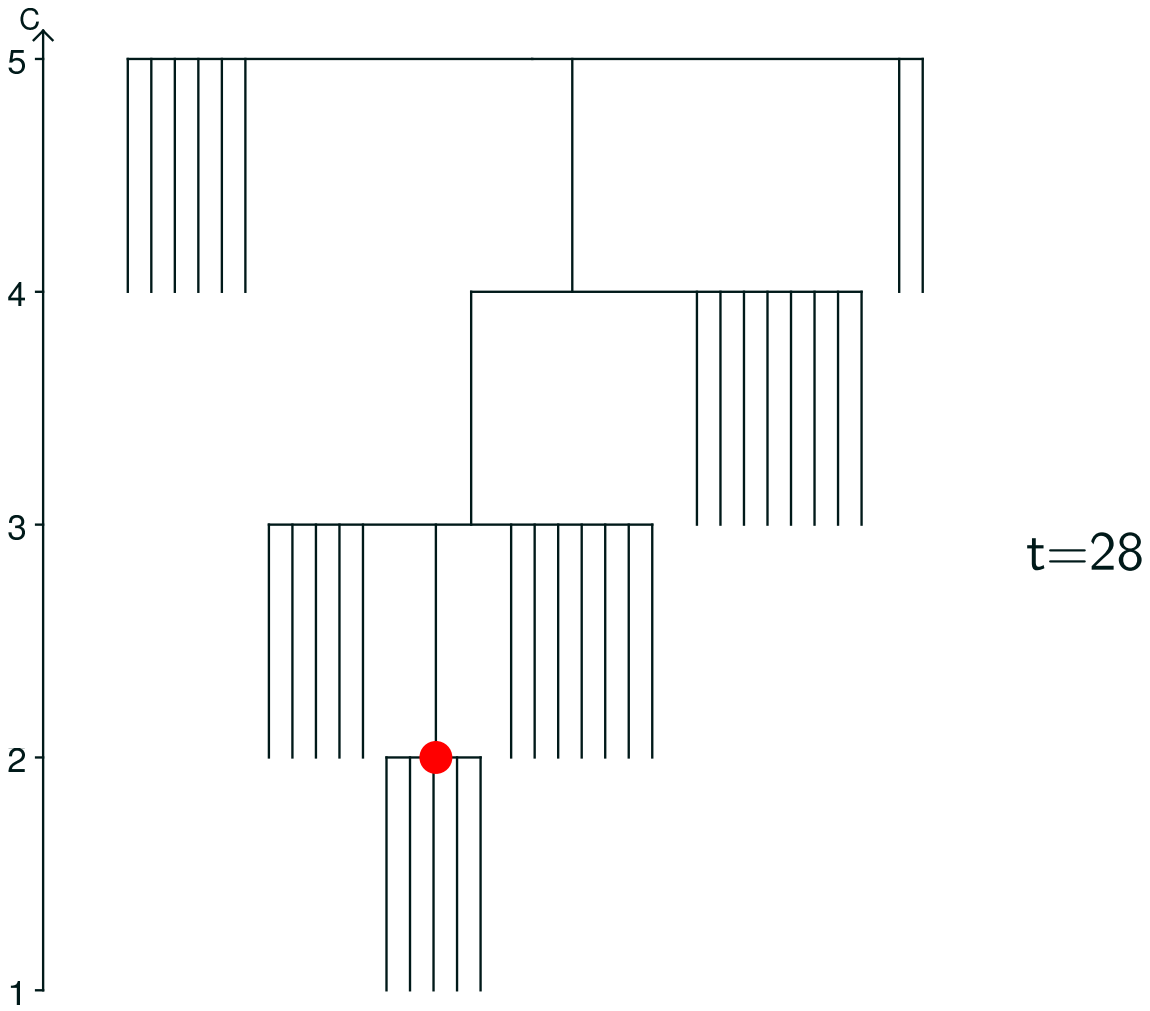
Simulated Annealing 40 Variables



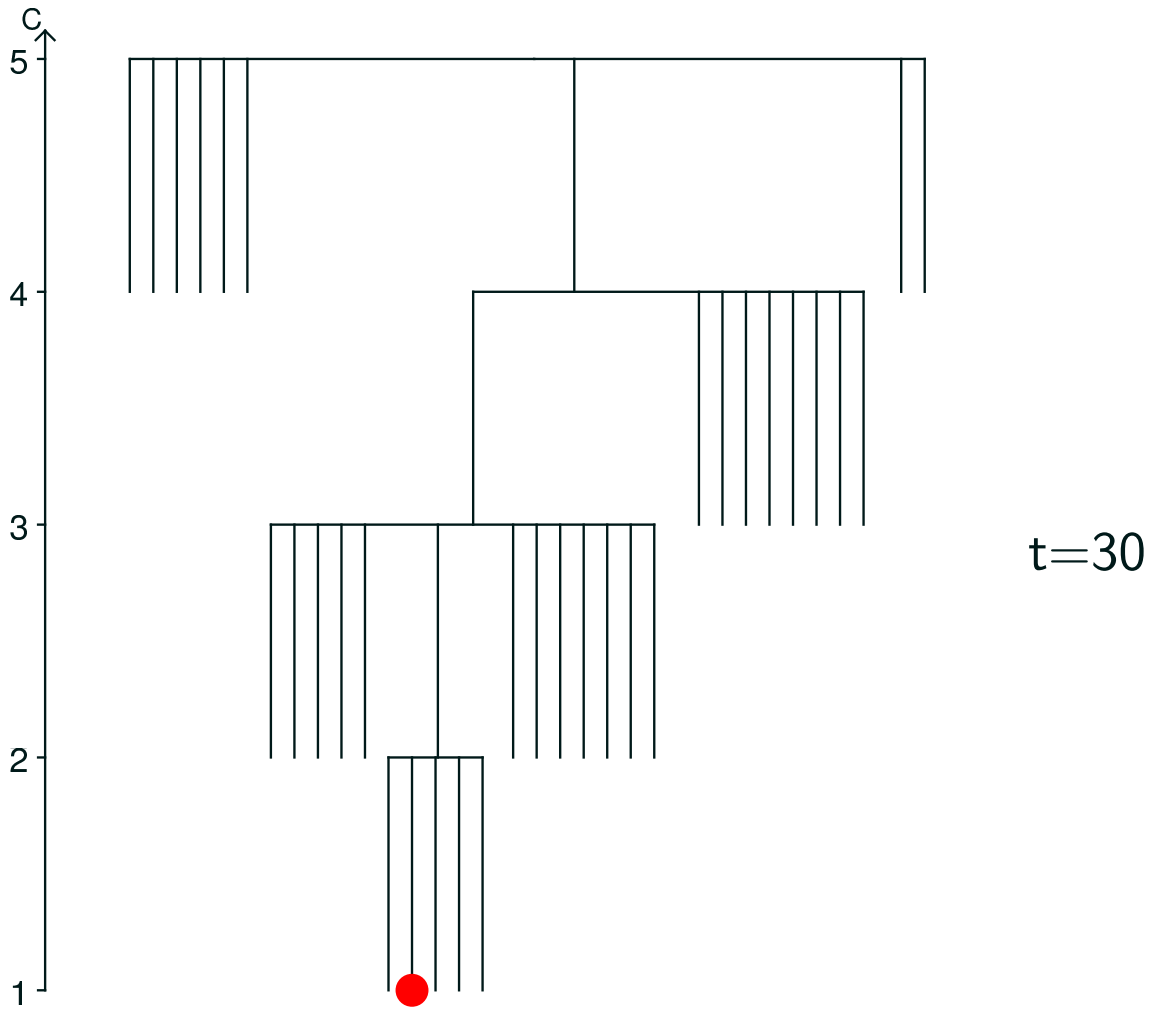
Simulated Annealing 40 Variables



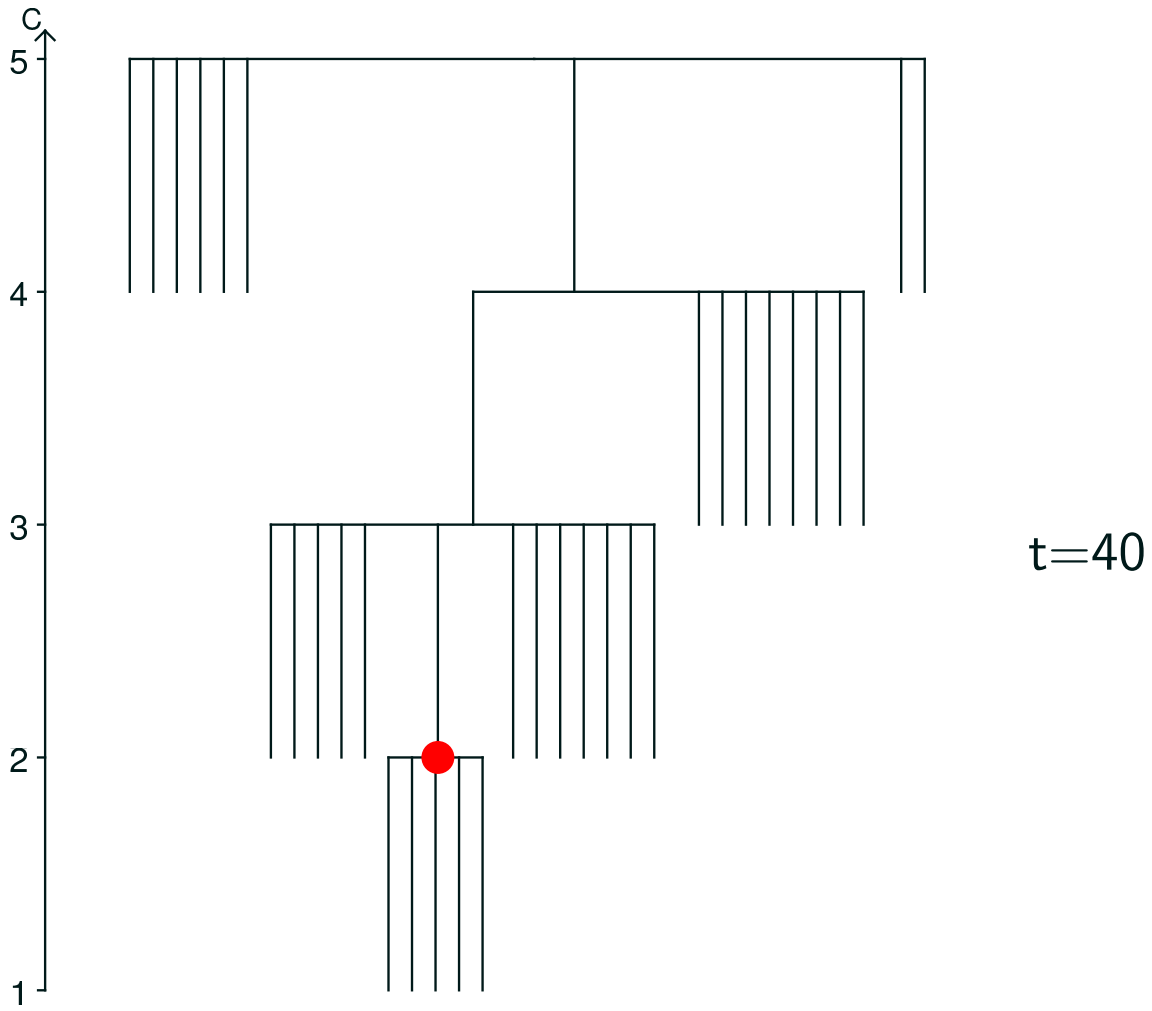
Simulated Annealing 40 Variables



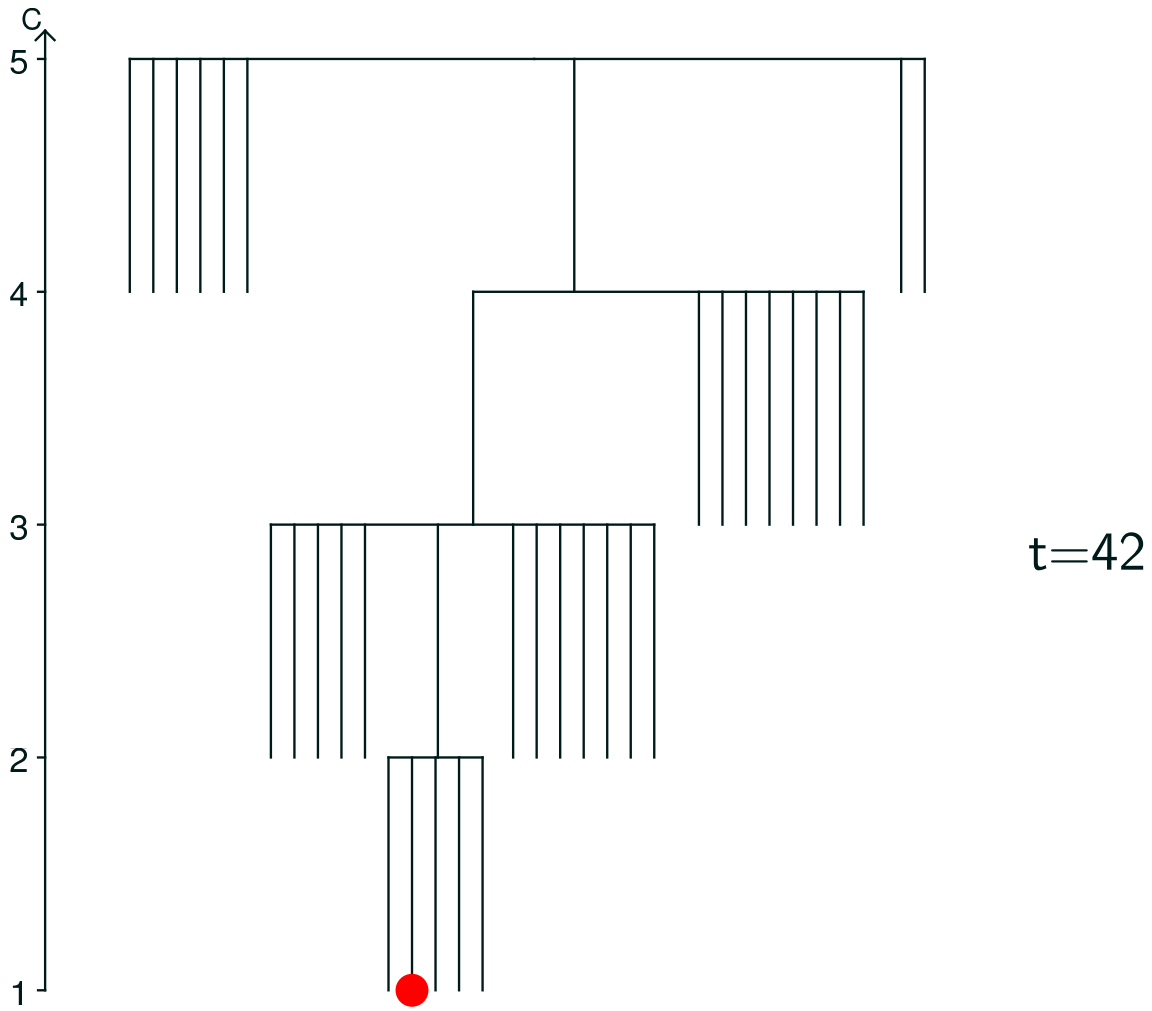
Simulated Annealing 40 Variables



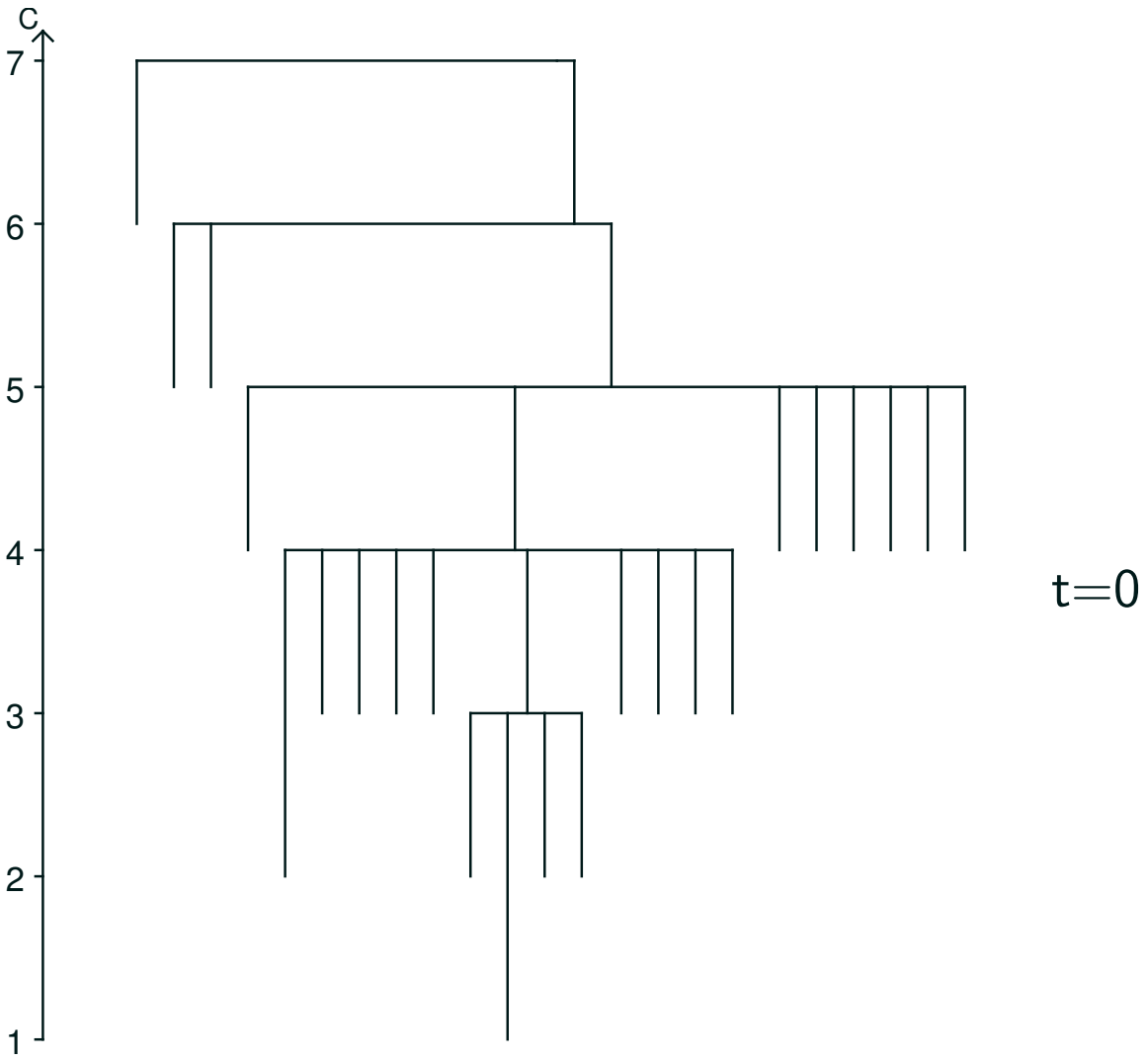
Simulated Annealing 40 Variables



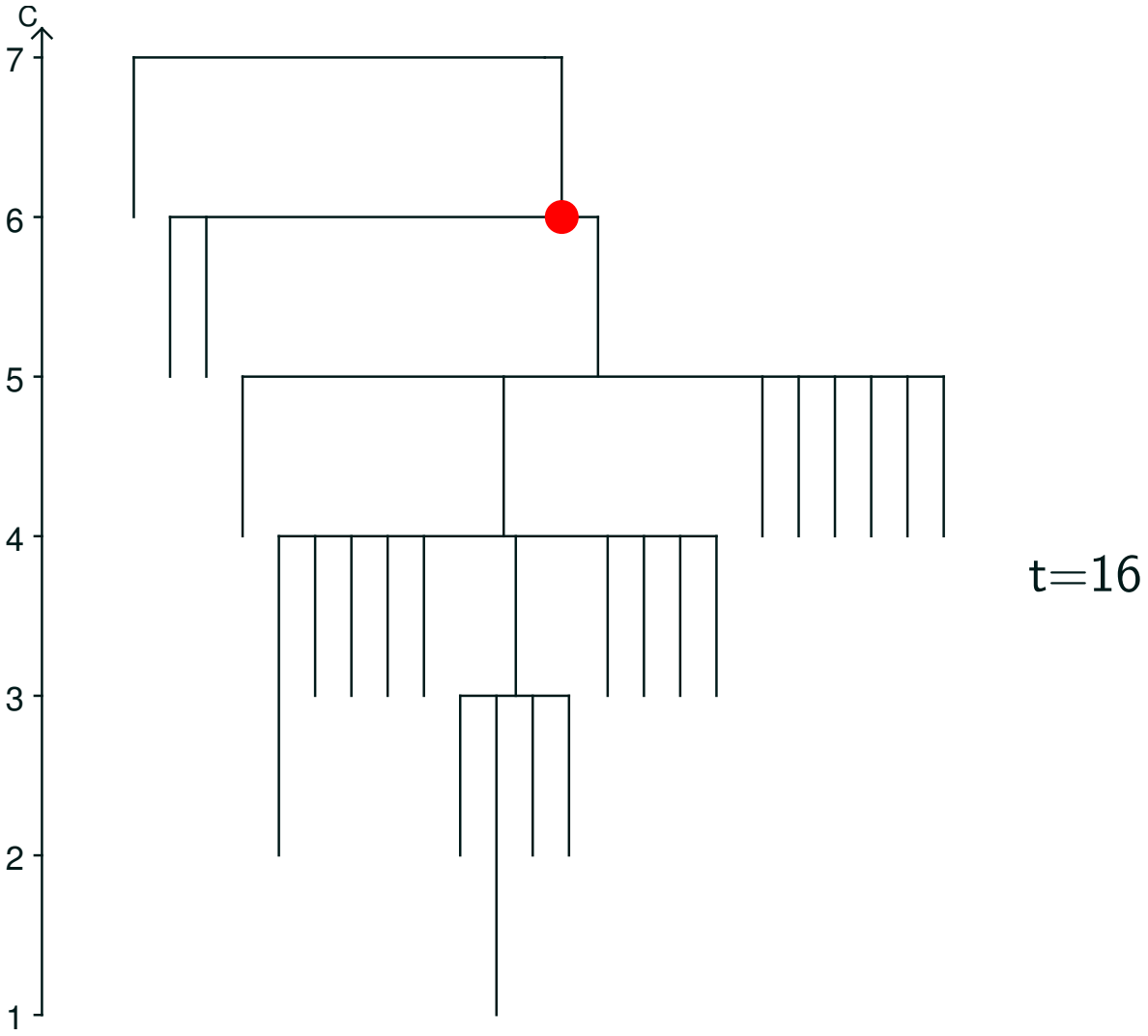
Simulated Annealing 40 Variables



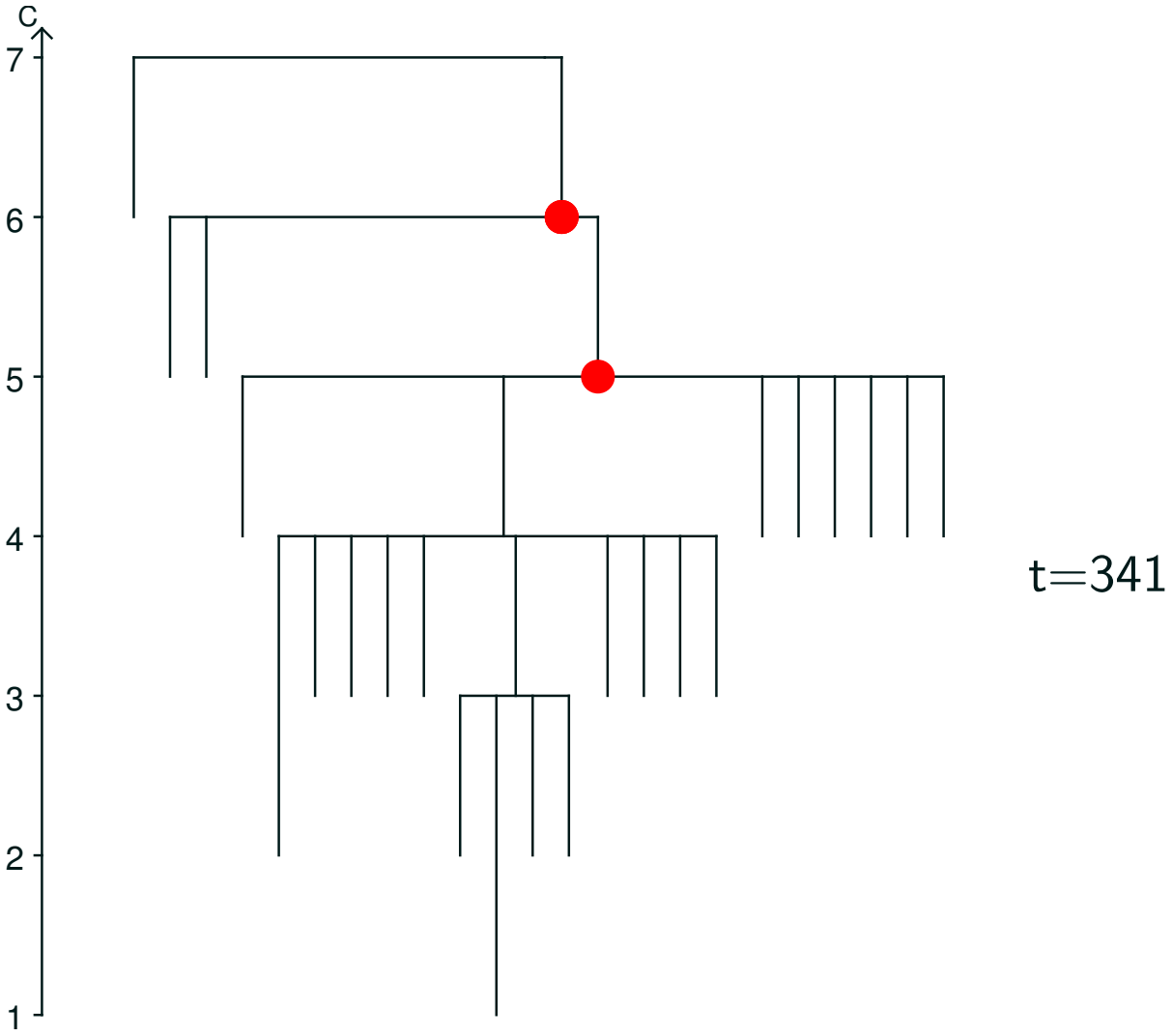
Genetic Algorithms 30 Variables



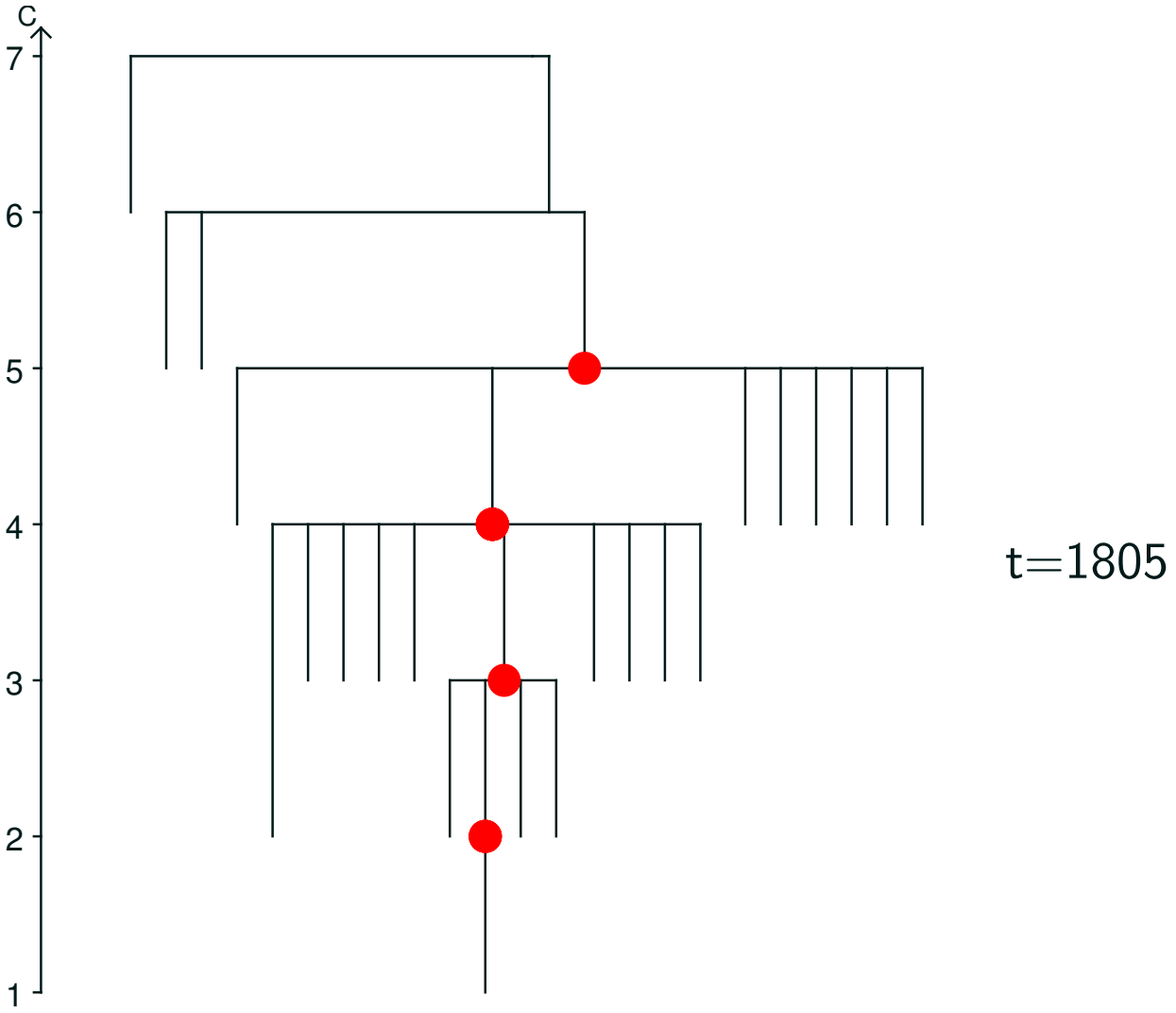
Genetic Algorithms 30 Variables



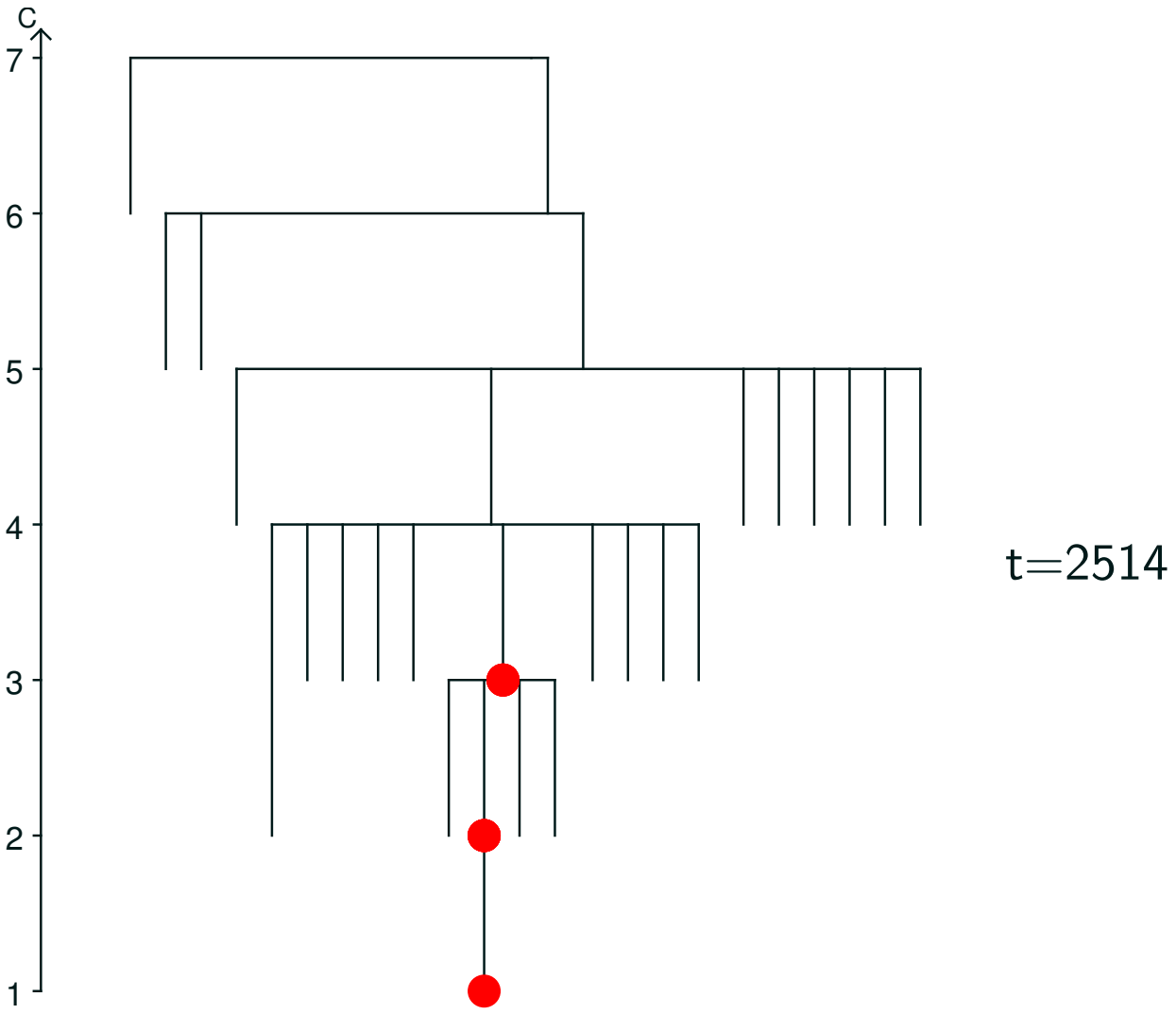
Genetic Algorithms 30 Variables



Genetic Algorithms 30 Variables

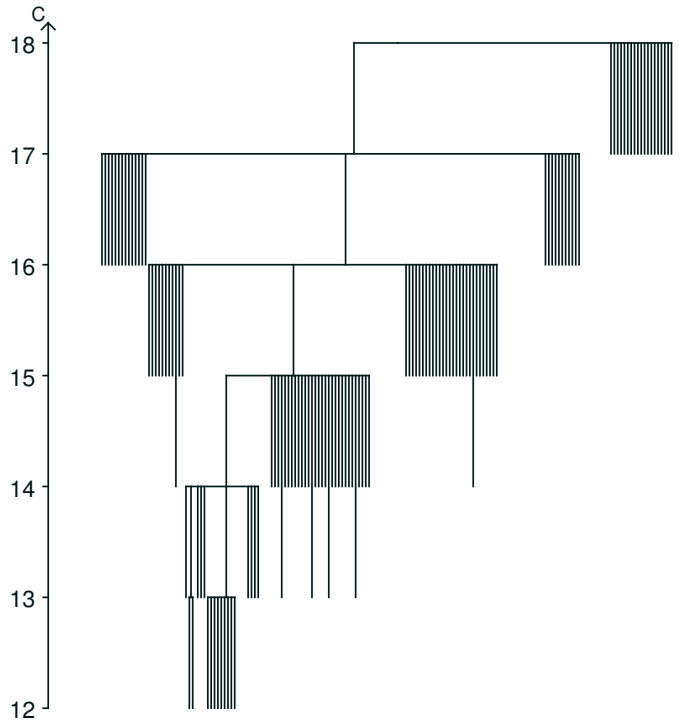


Genetic Algorithms 30 Variables



Outline

1. Optimisation Problems
2. Cost Landscapes
3. Barrier Trees
4. Example Instances
5. Mapping Configurations
6. Modelling the Problem



Modelling Optimisation Problems

- Model problems by amalgamating configurations
- Each node of on the barrier tree is a state
- Initial occupancy given by number of configurations in state
- Transition probabilities proportional to number of neighbouring pairs
- This is an approximation!
- Model has same structure of local minima and barriers as true problem

Modelling Optimisation Problems

- Model problems by amalgamating configurations
- Each node of on the barrier tree is a state
- Initial occupancy given by number of configurations in state
- Transition probabilities proportional to number of neighbouring pairs
- This is an approximation!
- Model has same structure of local minima and barriers as true problem

Modelling Optimisation Problems

- Model problems by amalgamating configurations
- Each node of on the barrier tree is a state
- Initial occupancy given by number of configurations in state
- Transition probabilities proportional to number of neighbouring pairs
- This is an approximation!
- Model has same structure of local minima and barriers as true problem

Modelling Optimisation Problems

- Model problems by amalgamating configurations
- Each node of on the barrier tree is a state
- Initial occupancy given by number of configurations in state
- Transition probabilities proportional to number of neighbouring pairs
- This is an approximation!
- Model has same structure of local minima and barriers as true problem

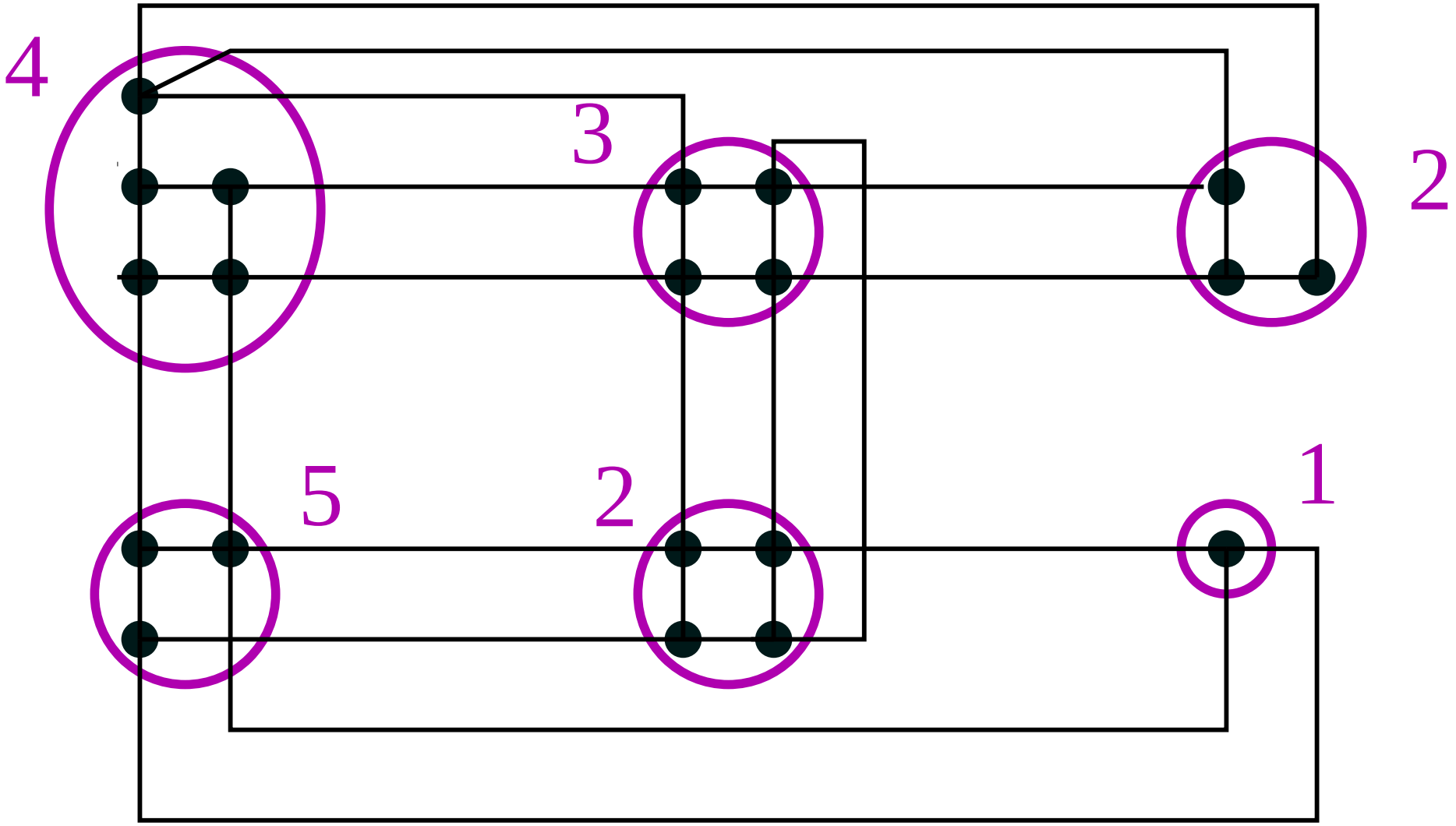
Modelling Optimisation Problems

- Model problems by amalgamating configurations
- Each node of on the barrier tree is a state
- Initial occupancy given by number of configurations in state
- Transition probabilities proportional to number of neighbouring pairs
- This is an approximation!
- Model has same structure of local minima and barriers as true problem

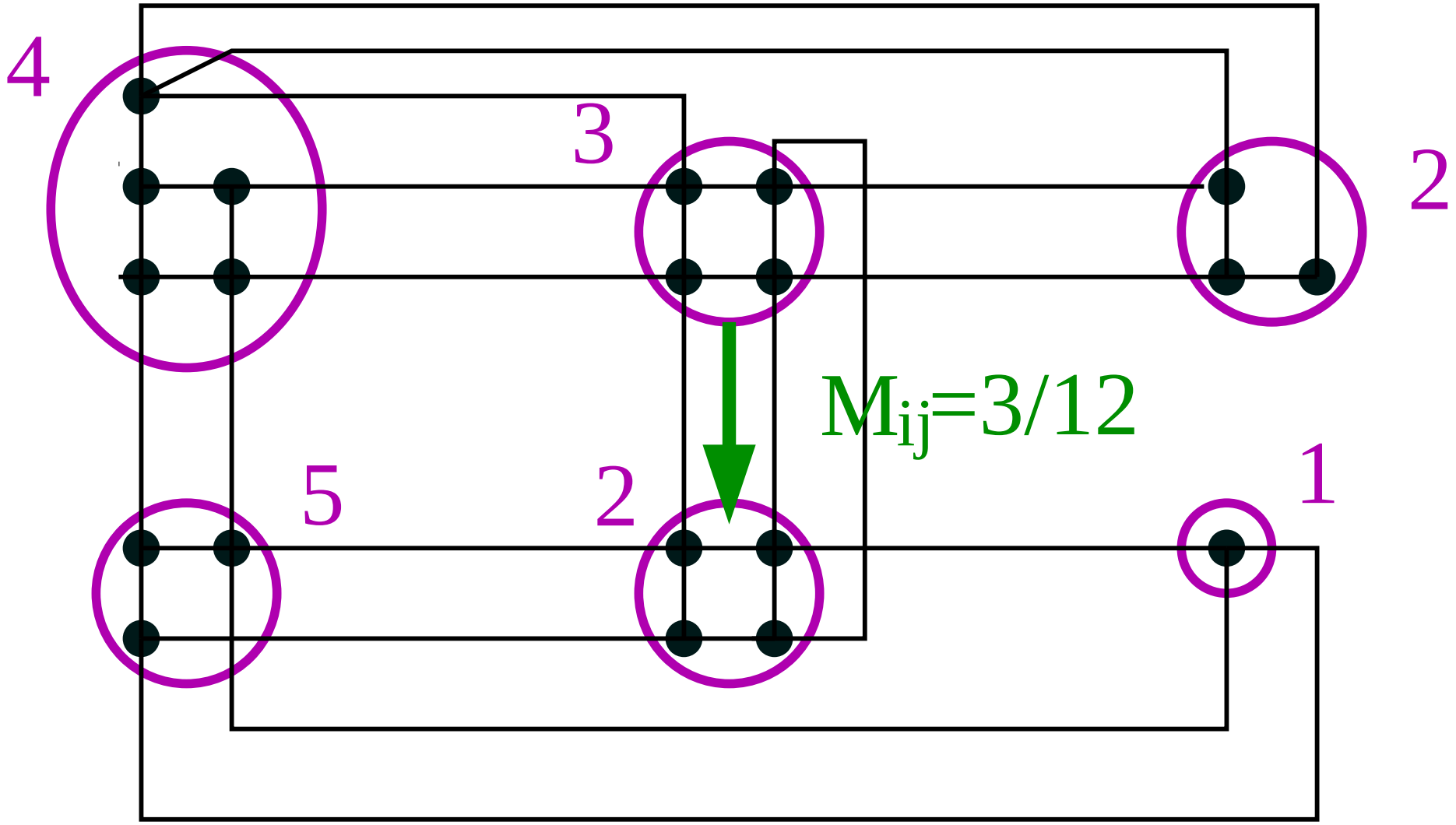
Modelling Optimisation Problems

- Model problems by amalgamating configurations
- Each node of on the barrier tree is a state
- Initial occupancy given by number of configurations in state
- Transition probabilities proportional to number of neighbouring pairs
- This is an approximation!
- Model has same structure of local minima and barriers as true problem

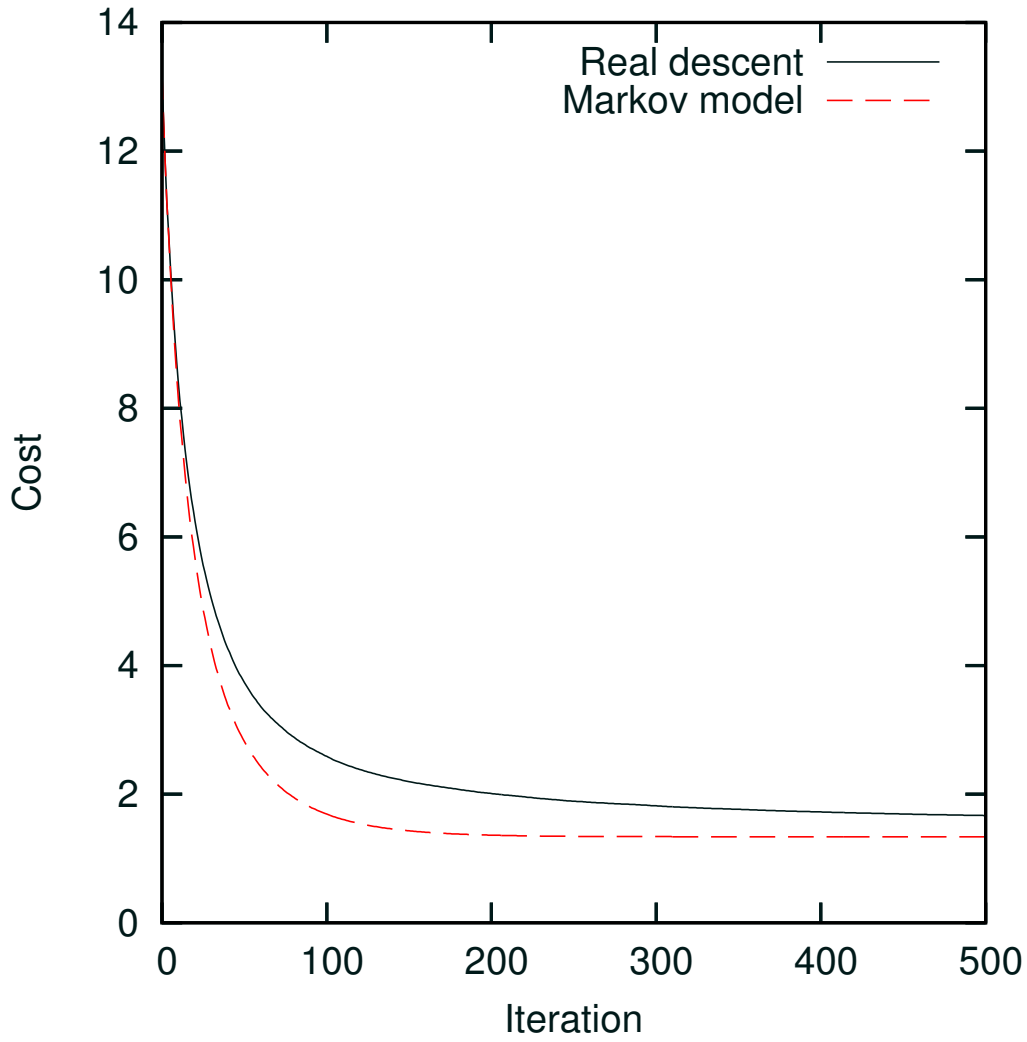
Schematic of Model Problem



Schematic of Model Problem

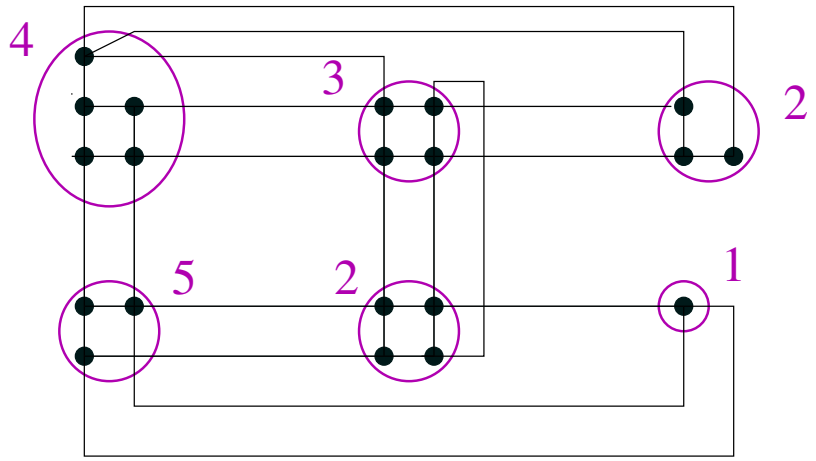


Descent: Max-3-Sat $N = 20$, $\alpha = 6$



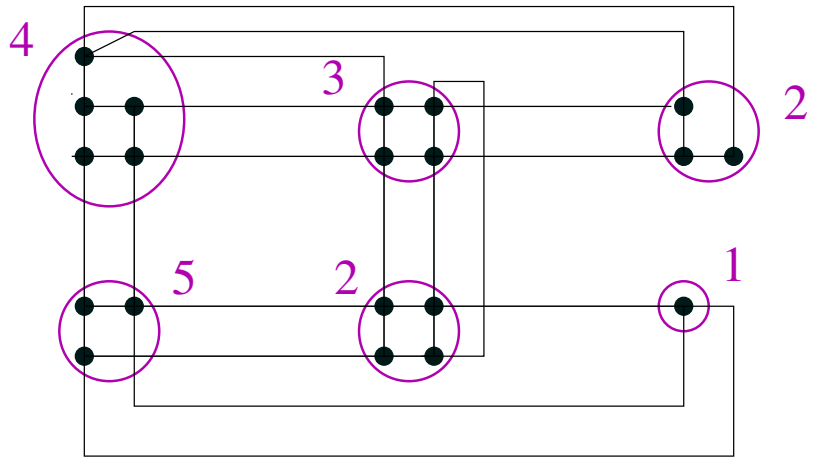
Comparison with True Problem

- Same qualitative behaviour
- Dynamics is slower—dynamics within state is not captured
- Model problems are systematically easier
- Configurations in state are not searched equally often



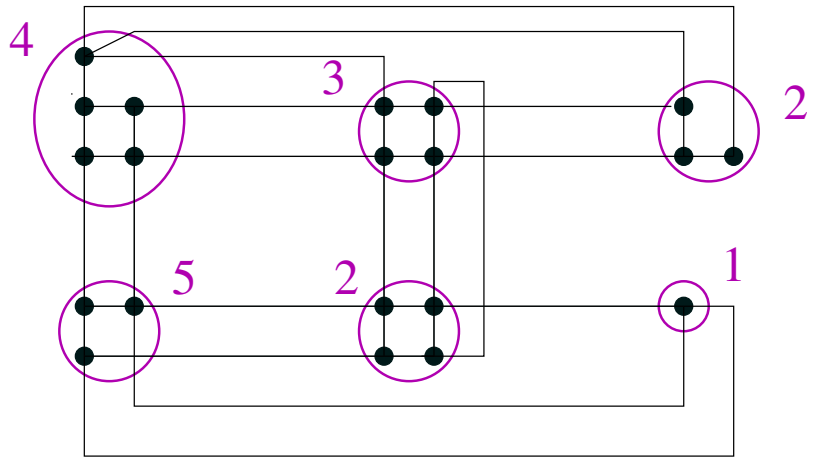
Comparison with True Problem

- Same qualitative behaviour
- Dynamics is slower—dynamics within state is not captured
- Model problems are systematically easier
- Configurations in state are not searched equally often



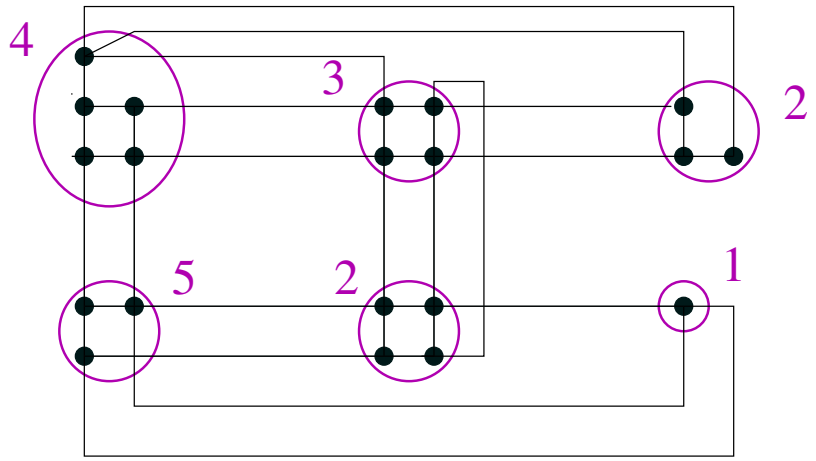
Comparison with True Problem

- Same qualitative behaviour
- Dynamics is slower—dynamics within state is not captured
- Model problems are systematically easier
- Configurations in state are not searched equally often



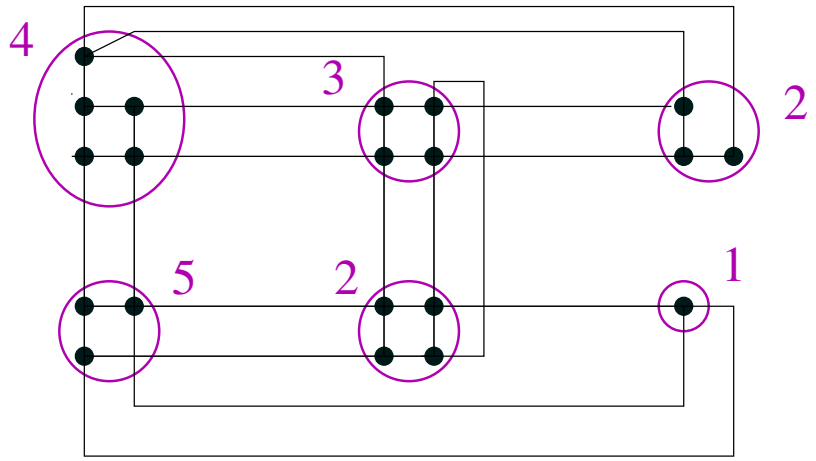
Comparison with True Problem

- Same qualitative behaviour
- Dynamics is slower—dynamics within state is not captured
- Model problems are systematically easier
- Configurations in state are not searched equally often

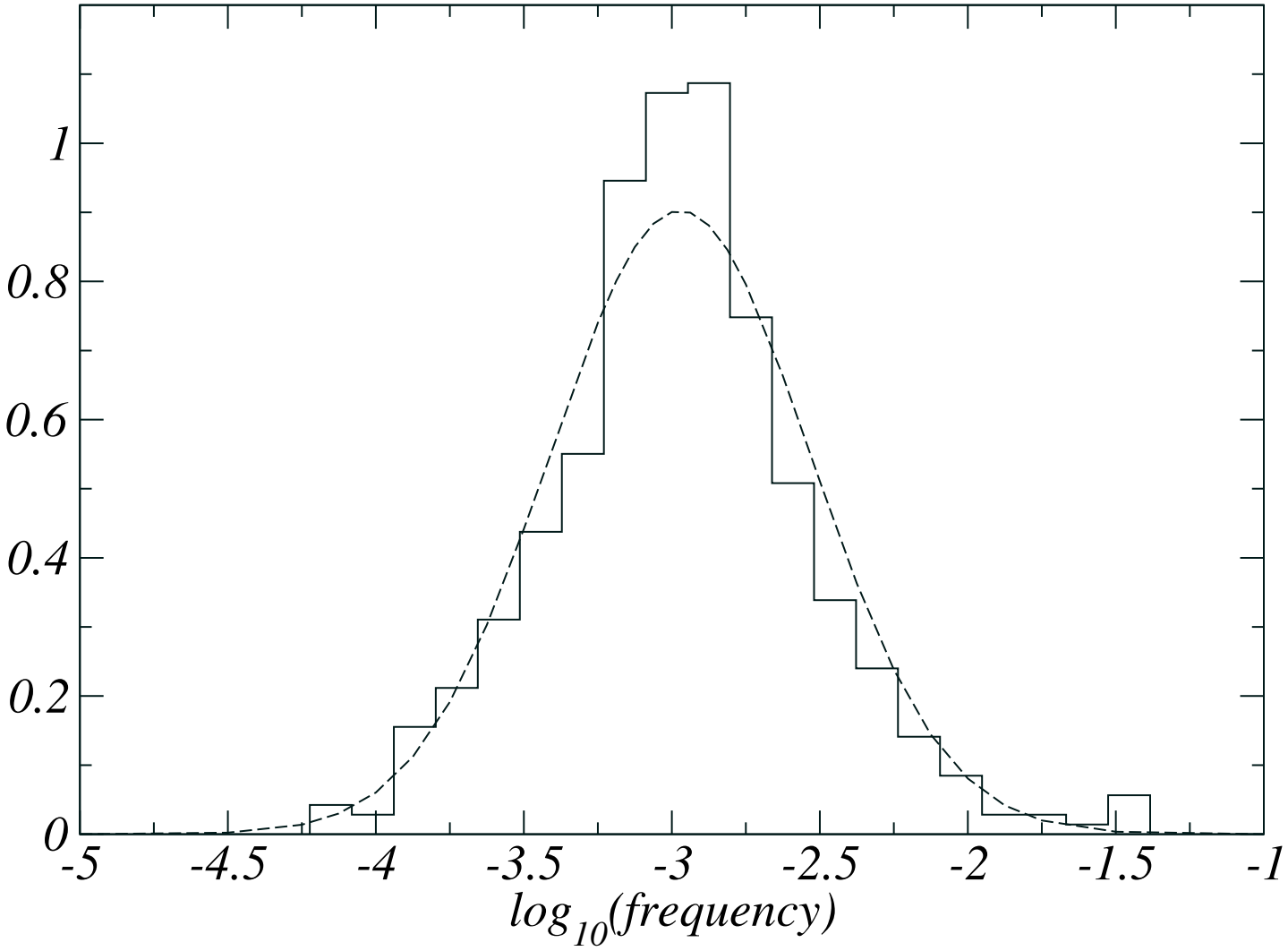


Comparison with True Problem

- Same qualitative behaviour
- Dynamics is slower—dynamics within state is not captured
- Model problems are systematically easier
- Configurations in state are not searched equally often



Frequency of Visiting Configurations



Configuration in a saddle-point level-connected set

Markov Models

- Many Heuristic search algorithms can be modelled as Markov processes

$$\boldsymbol{p}(t) = \mathbf{W}(t)\boldsymbol{p}(t-1)$$

- $\mathbf{W}(t)$ transition matrix
- $\mathbf{W}(t)$ depends on the connectivity matrix $\mathbf{M}$ and the heuristic
- For simulated annealing $\mathbf{W}(t)$ depends on the annealing temperature
- Because we have a small number of states this is tractable for moderate sized problems

Markov Models

- Many Heuristic search algorithms can be modelled as Markov processes

$$\boldsymbol{p}(t) = \mathbf{W}(t)\boldsymbol{p}(t-1)$$

- $\mathbf{W}(t)$ transition matrix
- $\mathbf{W}(t)$ depends on the connectivity matrix $\mathbf{M}$ and the heuristic
- For simulated annealing $\mathbf{W}(t)$ depends on the annealing temperature
- Because we have a small number of states this is tractable for moderate sized problems

Markov Models

- Many Heuristic search algorithms can be modelled as Markov processes

$$\boldsymbol{p}(t) = \mathbf{W}(t)\boldsymbol{p}(t-1)$$

- $\mathbf{W}(t)$ transition matrix
- $\mathbf{W}(t)$ depends on the connectivity matrix $\mathbf{M}$ and the heuristic
- For simulated annealing $\mathbf{W}(t)$ depends on the annealing temperature
- Because we have a small number of states this is tractable for moderate sized problems

Markov Models

- Many Heuristic search algorithms can be modelled as Markov processes

$$\boldsymbol{p}(t) = \mathbf{W}(t)\boldsymbol{p}(t-1)$$

- $\mathbf{W}(t)$ transition matrix
- $\mathbf{W}(t)$ depends on the connectivity matrix $\mathbf{M}$ and the heuristic
- For simulated annealing $\mathbf{W}(t)$ depends on the annealing temperature
- Because we have a small number of states this is tractable for moderate sized problems

Markov Models

- Many Heuristic search algorithms can be modelled as Markov processes

$$\boldsymbol{p}(t) = \mathbf{W}(t)\boldsymbol{p}(t-1)$$

- $\mathbf{W}(t)$ transition matrix
- $\mathbf{W}(t)$ depends on the connectivity matrix $\mathbf{M}$ and the heuristic
- For simulated annealing $\mathbf{W}(t)$ depends on the annealing temperature
- Because we have a small number of states this is tractable for moderate sized problems

Optimal Schedules

- We can compute optimum annealing schedule for the model analytically—has been studied previously on very small real problems
- Schedule depends on optimisation criteria
  - ⋆ Where-you-are—optimise the final result
  - ⋆ Best-so-far—optimise the best result obtained during run
- Can optimise for an ensemble of problems
- Can optimise other schedules such as the mutation rate for a descent algorithm

Optimal Schedules

- We can compute optimum annealing schedule for the model analytically—has been studied previously on very small real problems
- Schedule depends on optimisation criteria
  - ⋆ Where-you-are—optimise the final result
  - ⋆ Best-so-far—optimise the best result obtained during run
- Can optimise for an ensemble of problems
- Can optimise other schedules such as the mutation rate for a descent algorithm

Optimal Schedules

- We can compute optimum annealing schedule for the model analytically—has been studied previously on very small real problems
- Schedule depends on optimisation criteria
  - ⋆ Where-you-are—optimise the final result
  - ⋆ Best-so-far—optimise the best result obtained during run
- Can optimise for an ensemble of problems
- Can optimise other schedules such as the mutation rate for a descent algorithm

Optimal Schedules

- We can compute optimum annealing schedule for the model analytically—has been studied previously on very small real problems
- Schedule depends on optimisation criteria
  - ⋆ Where-you-are—optimise the final result
  - ⋆ Best-so-far—optimise the best result obtained during run
- Can optimise for an ensemble of problems
- Can optimise other schedules such as the mutation rate for a descent algorithm

Optimal Schedules

- We can compute optimum annealing schedule for the model analytically—has been studied previously on very small real problems
- Schedule depends on optimisation criteria
  - ⋆ Where-you-are—optimise the final result
  - ⋆ Best-so-far—optimise the best result obtained during run
- Can optimise for an ensemble of problems
- Can optimise other schedules such as the mutation rate for a descent algorithm

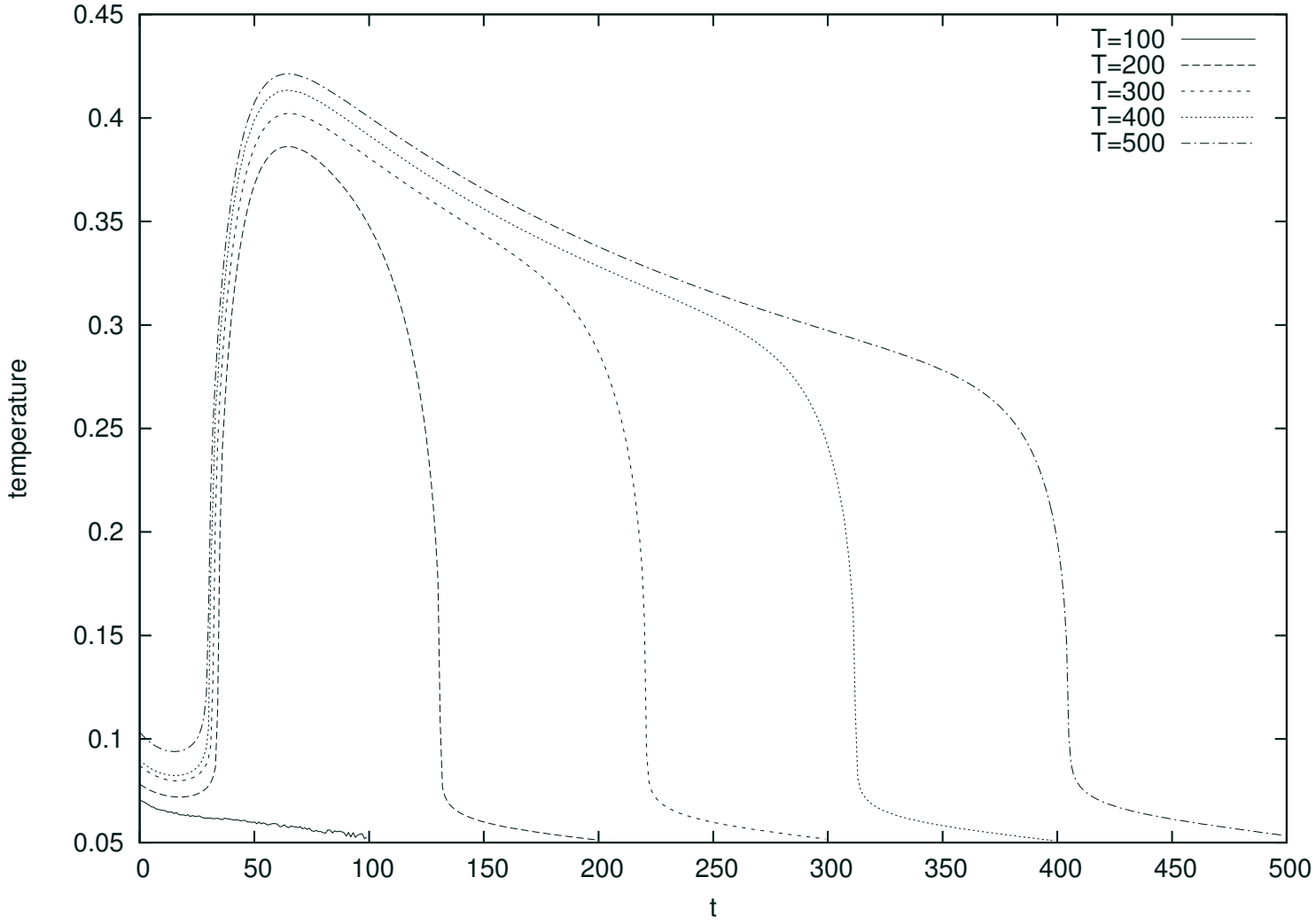
Optimal Schedules

- We can compute optimum annealing schedule for the model analytically—has been studied previously on very small real problems
- Schedule depends on optimisation criteria
  - ⋆ Where-you-are—optimise the final result
  - ⋆ Best-so-far—optimise the best result obtained during run
- Can optimise for an ensemble of problems
- Can optimise other schedules such as the mutation rate for a descent algorithm

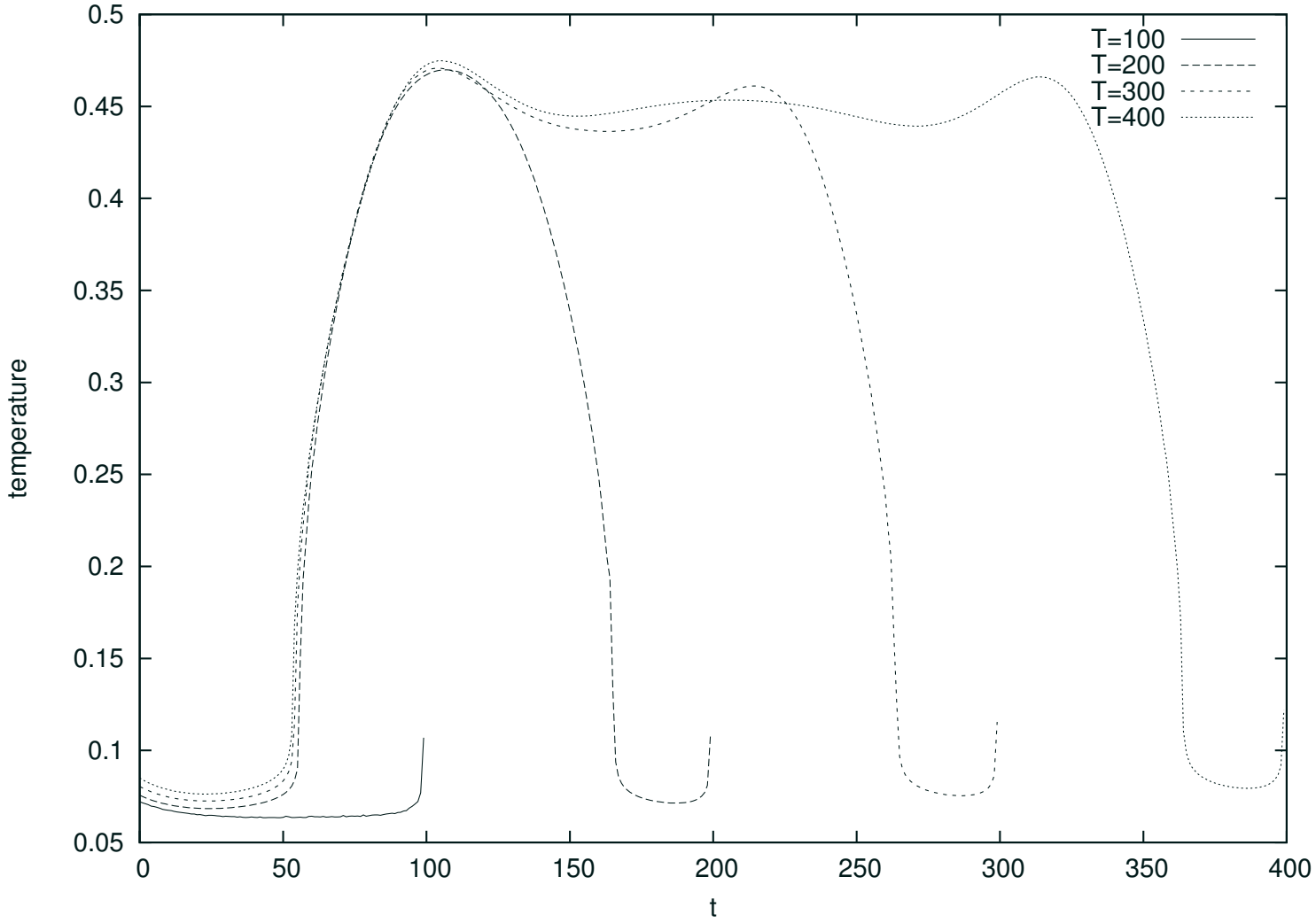
Optimal Schedules

- We can compute optimum annealing schedule for the model analytically—has been studied previously on very small real problems
- Schedule depends on optimisation criteria
  - ⋆ Where-you-are—optimise the final result
  - ⋆ Best-so-far—optimise the best result obtained during run
- Can optimise for an ensemble of problems
- Can optimise other schedules such as the mutation rate for a descent algorithm

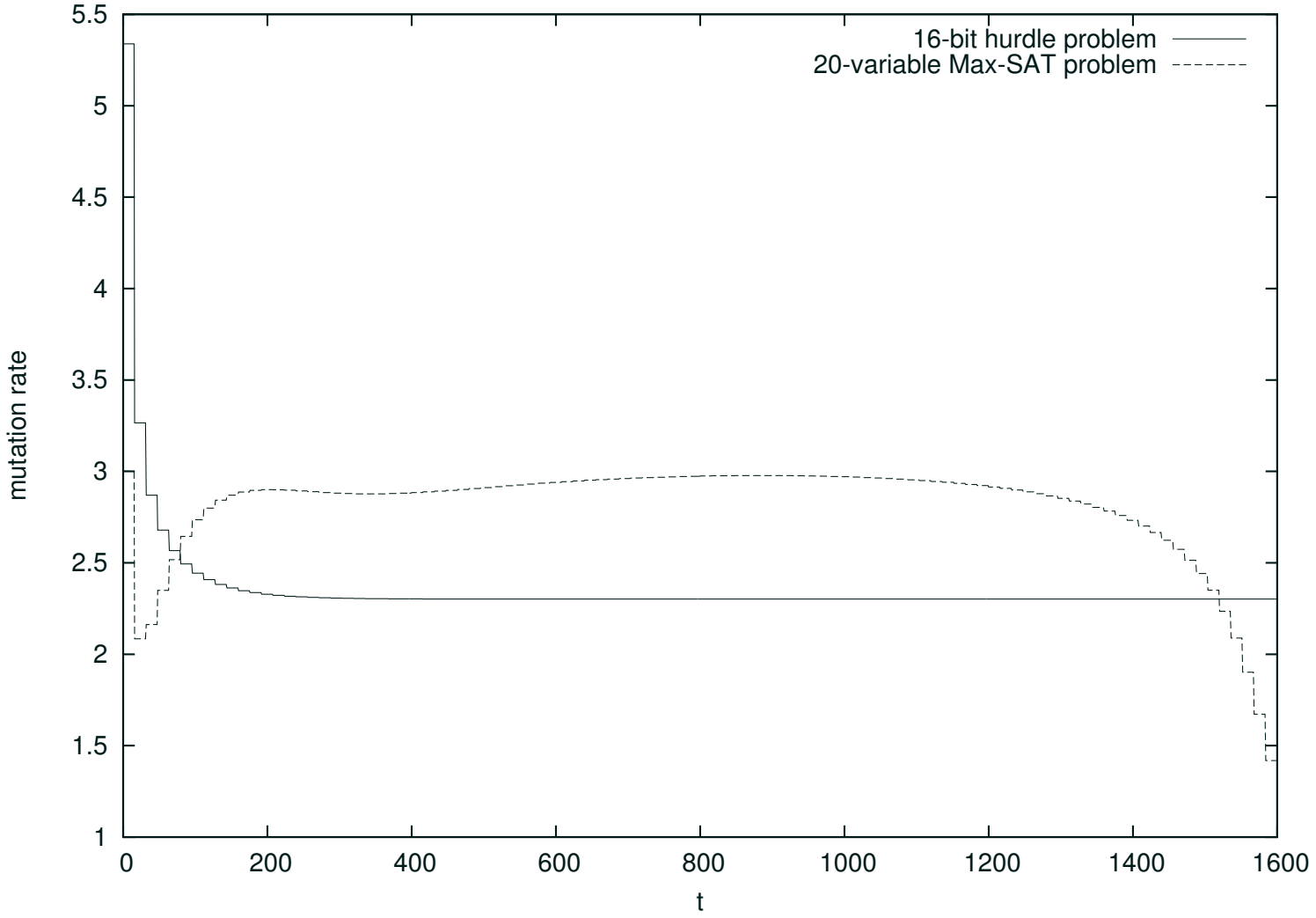
Max-Sat: Annealing Schedules with Where-You-Are



Max-Sat: Annealing Schedules with Best-So-Far



Max-Sat: Mutation Schedule with Where-You-Are



Conclusions

- Barrier trees provide a powerful tool for visualising cost landscapes
- By mapping configurations to nodes on the Barrier tree we can do much more
  - ⋆ measure more statistical properties
  - ⋆ visualise heuristic algorithms
  - ⋆ construct model problems
- Model problems open new possibilities for studying heuristic search

Conclusions

- Barrier trees provide a powerful tool for visualising cost landscapes
- By mapping configurations to nodes on the Barrier tree we can do much more
  - ⋆ measure more statistical properties
  - ⋆ visualise heuristic algorithms
  - ⋆ construct model problems
- Model problems open new possibilities for studying heuristic search

Conclusions

- Barrier trees provide a powerful tool for visualising cost landscapes
- By mapping configurations to nodes on the Barrier tree we can do much more
  - ⋆ measure more statistical properties
  - ⋆ visualise heuristic algorithms
  - ⋆ construct model problems
- Model problems open new possibilities for studying heuristic search

Conclusions

- Barrier trees provide a powerful tool for visualising cost landscapes
- By mapping configurations to nodes on the Barrier tree we can do much more
  - ⋆ measure more statistical properties
  - ⋆ visualise heuristic algorithms
  - ⋆ construct model problems
- Model problems open new possibilities for studying heuristic search

Conclusions

- Barrier trees provide a powerful tool for visualising cost landscapes
- By mapping configurations to nodes on the Barrier tree we can do much more
  - ⋆ measure more statistical properties
  - ⋆ visualise heuristic algorithms
  - ⋆ construct model problems
- Model problems open new possibilities for studying heuristic search

Conclusions

- Barrier trees provide a powerful tool for visualising cost landscapes
- By mapping configurations to nodes on the Barrier tree we can do much more
  - ⋆ measure more statistical properties
  - ⋆ visualise heuristic algorithms
  - ⋆ construct model problems
- Model problems open new possibilities for studying heuristic search

Any Questions?